# KARABÜK ÜNİVERSİTESİ
# BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ


## SENIOR PROJECT

# ImSafe
# Güvenli Resim Depolama ve Transfer Uygulaması

**Proje Öğrencileri**

**2015110205011 - Devran CANİKLİ**

**2016010205038 - Eren Alp ŞAKACI**

**Danışman**

**Dr. Öğr. Üyesi Burhan SELÇUK**

**2020**

## ÖZET

Günümüzde online iletişim uygulamalarının yaygınlaşması sebebiyle gün içerisinde yoğun bir şekilde resim alışverişi yapılmaktadır. Bu da çeşitli güvenlik ve gizlilik sorunlarına neden olmaktadır. Bu çalışmada hassas içeriğe sahip resimlerin simetrik bir anahtar ile şifrelenerek güvenliğinin sağlanması, bulut sunucuda depolanabilmesi, kullanıcılar arasında transfer edilebilmesi amaçlanmıştır. Simetrik şifreleme algoritması olarak Dr. Öğr. Üyesi Erdal Güvenoğlu'na ait "Resim Şifreleme Amacıyla Dinamik S Kutusu Tasarımı İçin Bir Yöntem" adlı çalışmada bulunan algoritma kullanılmıştır [1]. Bu algoritma, resmin RGB değerlerinin, kullanıcı tarafından girilen seed (anahtar) değerine bağlı olarak oluşturulan rastgele sayıların yardımıyla değiştirilmesi yöntemine dayanmaktadır. Şifrelenen resim sadece anahtarı bilen kullanıcı tarafından çözülebilmektedir. Bu sayede resmin paylaşıldığı ortama saldırılması halinde üçüncü şahıslar resmin içeriğini görüntüleyememektedir. Şifreleme aşamasında şifrelenecek resim parçalara ayrılarak paralel olarak şifrelenir ve birleştirilir. Ayrıca şifrelenen resim RSA dijital imza algoritması ile imzalanır. Şifrelenmiş resmi çözme aşamasında ise kullanıcının e-imzası doğrulanıp, doğru anahtarın girilmesi halinde şifrelenmiş resim orijinal haline geri döndürülür. Bulut sunucu üzerinde çalışan Django RESTful web servis ve bu servisi kullanan Android mobil uygulama geliştirildi. Kullanıcılar bu uygulamayı kullanarak resimlerini şifreleyip bulut üzerinde depolayabilmekte ve bu resimleri kendi aralarında transfer edebilmektedir.

## 1. GİRİŞ

AES (Advanced Encryption Standard) veya DES (Data Encryption Standard) gibi blok şifreleme algoritmalarının, resim dosyalarının boyutlarının büyük olmasından dolayı resim şifreleme için kullanılması uygun görülmemektedir [1]. Bu çalışmada AES'in temelinde bulunan S-box (substitution-box) yöntemi kullanılmıştır. S-box, simetrik şifreleme algoritmalarında yerine koyma (substitution) amacıyla kullanılan bir tekniktir. Blok şifrelemede kullanılan S-box'lar genellikle statiktir. Bu çalışmada kullanılan S-box, güvenliği artırmak amacıyla belirlenen anahtara göre dinamik bir şekilde üretilmiştir.

## 2. RESİM ŞİFRELEME VE ÇÖZME İŞLEMLERİ

### 2.1. S-box Üretimi

Çalışmada kullanılan S-box 0-255 aralığında 256 adet sayı içermektedir. Bu sayıları 16x16 boyutundaki matrise dinamik (rastgele) olarak yerleştirebilmek için öncelikle bir seed (anahtar) değeri belirlenmiştir. Bu anahtar, Knuth Shuffle algoritmasında sayıları karıştırmak amacıyla kullanılmıştır [2]. Daha sonra bu algoritmasının kullanılmasıyla karıştırılan sayılar on altılık sayı sistemine dönüştürülür.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 19 | B6 | 15 | AF | 42 | 06 | 01 | EF | 3D | 78 | 9A | 7F | 30 | 07 | 60 | DF |
| 1 | 96 | 46 | B4 | A8 | F5 | ED | 02 | DE | C0 | 89 | BC | 7A | DD | 5C | 18 | 82 |
| 2 | 09 | AD | 03 | E7 | 95 | C6 | 2F | D7 | 72 | 00 | 6C | D4 | FE | E3 | 08 | 1C |
| 3 | 67 | 38 | B0 | F7 | EA | 6E | 39 | B5 | 66 | 2D | 94 | F0 | C8 | 2B | 32 | 6B |
| 4 | 16 | 84 | 77 | FC | A6 | 93 | C1 | 2C | 14 | 3E | 70 | B8 | E9 | F1 | 69 | 57 |
| 5 | 59 | D2 | E2 | 74 | 65 | D0 | BE | 7C | F6 | A4 | A0 | 5A | 81 | 8A | C4 | 29 |
| 6 | F8 | 4A | 8E | 5F | C3 | 86 | 61 | FA | 68 | F3 | 28 | E4 | 44 | F4 | 41 | 35 |
| 7 | 05 | 11 | BD | 6D | 1B | 12 | 5D | 8C | 34 | B9 | C9 | A1 | 27 | 1F | 9F | 22 |
| 8 | 55 | EB | 62 | 52 | 75 | F2 | E1 | 31 | DA | 5E | 1D | 6A | 8F | D5 | 49 | A3 |
| 9 | D1 | 0D | 0C | 91 | CB | 98 | CF | 33 | CE | EE | 0E | 76 | FB | B2 | 04 | 87 |
| 10 | CD | 43 | 5B | 20 | 92 | 99 | F9 | D3 | 8B | 4E | 83 | E0 | 3B | C5 | 50 | 3F |
| 11 | E6 | FD | 79 | 48 | A9 | A2 | D8 | 9E | 47 | CA | 1E | 10 | 6F | DB | 8D | BA |
| 12 | 36 | 9D | 2E | 3C | 25 | 0B | EC | 85 | C7 | AB | 4F | 54 | 13 | 2A | 9B | BB |
| 13 | BF | E8 | C2 | 51 | 3A | D9 | 90 | 58 | AA | 4C | 9C | D6 | 7D | 7E | A7 | DC |
| 14 | 24 | 0A | 40 | B3 | 26 | 4D | B7 | E5 | 7B | 21 | 23 | 71 | B1 | 0F | 63 | 53 |
| 15 | 64 | 97 | AC | 45 | 17 | FF | AE | 56 | 73 | CC | 37 | 88 | 80 | 4B | A5 | 1A |

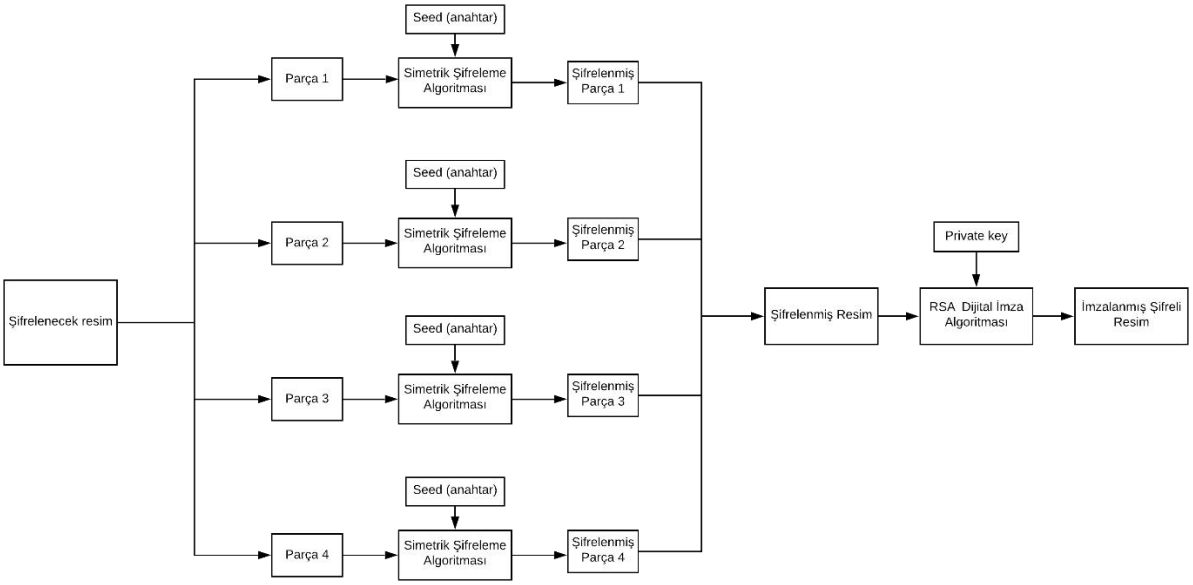
*Şekil 2.1. '123' Seed Değerine Göre Üretilen S-box*

Ardından sadece çözme aşamasında kullanılan ters S-box üretilir. Ters S-box, S-box değerlerine bağlı üretilen bir matristir. Örneğin, Şekil 2.1'de bulunan S-box matrisinde 3. satır 2. sütundaki değer B0'dır. Ters S-box'ta ise B. Satır (11. Satır) 0. Sütundaki değer '32' olacaktır.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 29 | 06 | 16 | 22 | 9E | 70 | 05 | 0D | 2E | 20 | E1 | C5 | 92 | 91 | 9A | ED |
| 1 | BB | 71 | 75 | CC | 48 | 02 | 40 | F4 | 1E | 00 | FF | 74 | 2F | 8A | BA | 7D |
| 2 | A3 | E9 | 7F | EA | E0 | C4 | E4 | 7C | 6A | 5F | CD | 3D | 47 | 39 | C2 | 26 |
| 3 | 0C | 87 | 3E | 97 | 78 | 6F | C0 | FA | 31 | 36 | D4 | AC | C3 | 08 | 49 | AF |
| 4 | E2 | 6E | 04 | A1 | 6C | F3 | 11 | B8 | B3 | 8E | 61 | FD | D9 | E5 | A9 | CA |
| 5 | AE | D3 | 83 | EF | CB | 80 | F7 | 4F | D7 | 50 | 5B | A2 | 1D | 76 | 89 | 63 |
| 6 | 0E | 66 | 82 | EE | F0 | 54 | 38 | 30 | 68 | 4E | 8B | 3F | 2A | 73 | 35 | BC |
| 7 | 4A | EB | 28 | F8 | 53 | 84 | 9B | 42 | 09 | B2 | 1B | E8 | 57 | DC | DD | 0B |
| 8 | FC | 5C | 1F | AA | 41 | C7 | 65 | 9F | FB | 19 | 5D | A8 | 77 | BE | 62 | 8C |
| 9 | D6 | 93 | A4 | 45 | 3A | 24 | 10 | F1 | 95 | A5 | 0A | CE | DA | C1 | B7 | 7E |
| 10 | 5A | 7B | B5 | 8F | 59 | FE | 44 | DE | 13 | B4 | D8 | C9 | F2 | 21 | F6 | 03 |
| 11 | 32 | EC | 9D | E3 | 12 | 37 | 01 | E6 | 4B | 79 | BF | CF | 1A | 72 | 56 | D0 |
| 12 | 18 | 46 | D2 | 64 | 5E | AD | 25 | C8 | 3C | 7A | B9 | 94 | F9 | A0 | 98 | 96 |
| 13 | 55 | 90 | 51 | A7 | 2B | 8D | DB | 27 | B6 | D5 | 88 | BD | DF | 1C | 17 | 0F |
| 14 | AB | 86 | 52 | 2D | 6B | E7 | B0 | 23 | D1 | 4C | 34 | 81 | C6 | 15 | 99 | 07 |
| 15 | 3B | 4D | 85 | 69 | 6D | 14 | 58 | 33 | 60 | A6 | 67 | 9C | 43 | B1 | 2C | F5 |

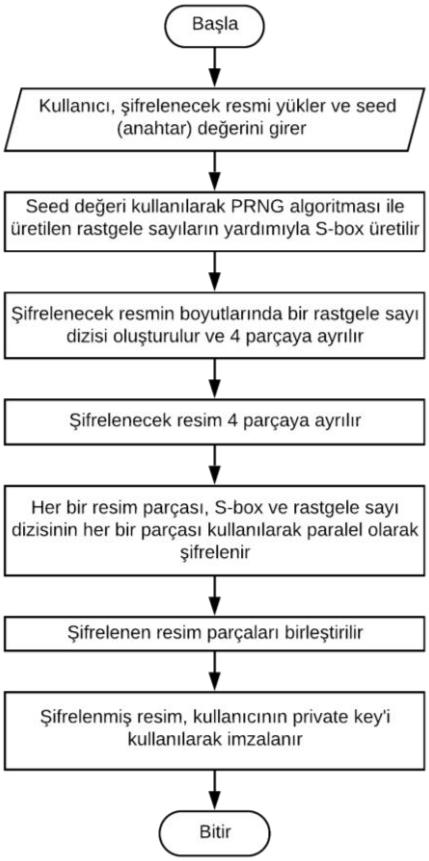
*Şekil 2.2. '123' Seed Değerine Göre Üretilen Ters S-box*

## 2.2. Şifreleme Aşaması




*Şekil 2.3. Şifreleme Aşaması*

Şekil 2.3'te görüldüğü üzere, şifrelenecek resim 4 parçaya ayrılır. Daha sonra her bir parça paralel olarak simetrik şifreleme algoritmasına sokulup, kullanıcının girdiği seed (anahtar) değeri kullanılarak şifrelenir. Şifrelenmiş parçalar birleştirilir ve orijinal resmin boyutlarında şifrelenmiş resim elde edilir. Şifrelenmiş resim RSA dijital imza algoritmasına sokulup, kullanıcının private key'i kullanılarak imzalanır. Sonuç olarak imzalanmış şifreli resim elde edilir. Şifreleme aşamasının akış diyagramı aşağıdaki gibidir:




*Şekil 2.4. Şifreleme Aşamasının Akış Diyagramı*

Şekil 2.4'te anlatıldığı üzere, resmi şifreleyecek olan kullanıcı, şifrelenecek olan resmi yükler ve belirlediği seed (anahtar) değerini girer. Girilen seed değerine göre PRNG (Pseudo-Random Number Generator) algoritması kullanılarak rastgele sayı üretimi başlar. Oluşturulan rastgele sayılar kullanılarak S-box matrisinde bulunan sayılar karıştırılır ve S-box üretimi sonlandırılır. Şifrelenecek resmin boyutlarında bir rastgele sayı dizisi oluşturulur ve 4 parçaya ayrılır. Ayrıca şifrelenecek resim de 4 parçaya ayrılır. Daha sonra multiprocessing yöntemi kullanılarak 4 ayrı process oluşturulur. Bu process'lere her bir resim parçası, S-box ve rastgele sayı dizisinin her bir parçası sokulur. Ardından bütün parçalar simetrik şifreleme algoritması kullanılarak paralel olarak şifrelenir. Şifrelenen parçalar birleştirilir ve kullanıcının private key'i kullanılarak RSA dijital imza algoritması ile imzalanır.

```python
def encrypt(obj, password):
    img = cv2.imread(obj.image.path)

    height = int(len(img))
    width = int(len(img[0]))

    array_slicer = Slicer(img, height, width)
    img_top_left, img_top_right, img_bottom_left, img_bottom_right = array_slicer.slice()

    np.random.seed(int(password))

    shuffle = KnuthShuffle()
    s_box = shuffle.create_s_box(np.random)

    random_numbers = np.random.randint(0, 16, (height, width, 6))
    array_slicer.set_array(random_numbers)
    rand_top_left, rand_top_right, rand_bottom_left, rand_bottom_right = array_slicer.slice()

    image_encryption = ImageEncryption()

    start = time.perf_counter()

    result_queue = Queue()

    procs = []
    proc1 = Process(target=image_encryption.encrypt, args=(s_box, rand_top_left, img_top_left, result_queue, 1))
    procs.append(proc1)
    proc1.start()

    proc2 = Process(target=image_encryption.encrypt, args=(s_box, rand_top_right, img_top_right, result_queue, 2))
    procs.append(proc2)
    proc2.start()

    proc3 = Process(target=image_encryption.encrypt, args=(s_box, rand_bottom_left, img_bottom_left, result_queue, 3))
    procs.append(proc3)
    proc3.start()

    proc4 = Process(target=image_encryption.encrypt, args=(s_box, rand_bottom_right, img_bottom_right, result_queue, 4))
    procs.append(proc4)
    proc4.start()

    image_slice_list = [result_queue.get() for i in range(4)]
    image_slice_list.sort(key=Util.sort_second)

    for proc in procs:
        proc.join()

    finish = time.perf_counter()

    print('Finished in {} second(s)'.format(finish - start))

    encrypted_image = Slicer.concatenate(image_slice_list[0][0], image_slice_list[1][0], image_slice_list[2][0],
                                         image_slice_list[3][0])

    media_root = settings.MEDIA_ROOT
    write_path = os.path.join(settings.MEDIA_ROOT, 'uploads/'+obj.name+str(obj.id)+'.png')
    cv2.imwrite(write_path, encrypted_image)
    os.remove(obj.image.path)
    obj.image = 'uploads/'+obj.name+str(obj.id)+'.png'

    return True
```

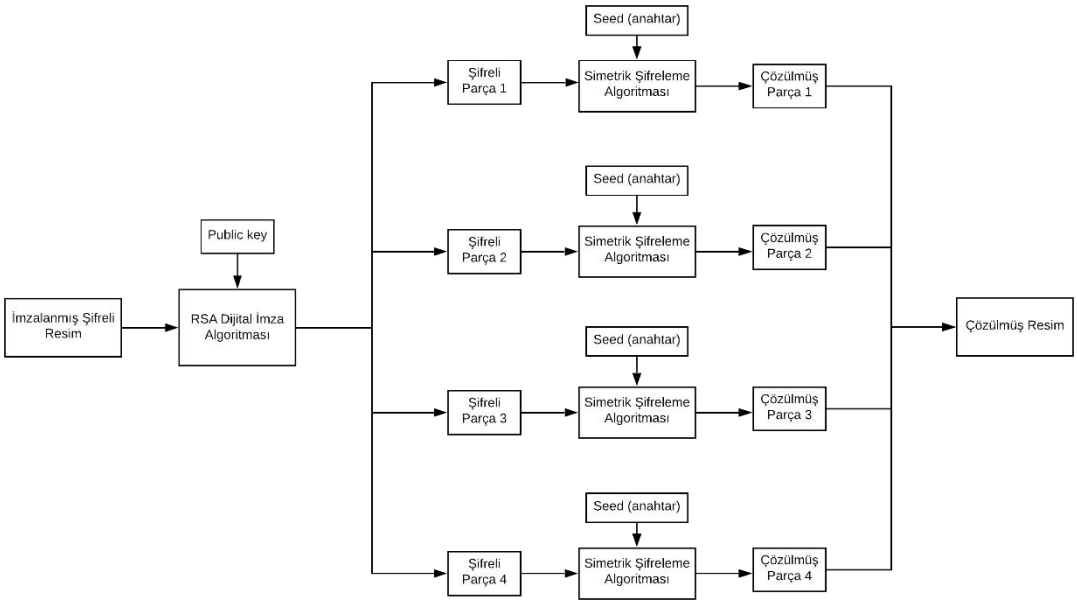
*Şekil 2.5. Process'lerin Oluşturulduğu Şifreleme Metodu*

Şekil 2.5'te process'lerin oluşturulduğu şifreleme metodu görülmektedir. Bu metot ayrıca resim parçalama, S-box oluşturma, resim parçalarını birleştirme ve Şekil 2.6'da görülen şifreleme algoritmasının çalıştırıldığı metotları çağırır.

```
def encrypt(self, s_box, random_numbers, im, result_queue, image_id):
    for i in range(len(im)):
        for j in range(len(im[0])):
            b, g, r = im[i][j]

            hex_r = Util.convert_dec_to_hex(r)
            hex_g = Util.convert_dec_to_hex(g)
            hex_b = Util.convert_dec_to_hex(b)

            row_r = int(hex_r[0], 16)
            column_r = int(hex_r[1], 16)

            row_g = int(hex_g[0], 16)
            column_g = int(hex_g[1], 16)

            row_b = int(hex_b[0], 16)
            column_b = int(hex_b[1], 16)

            new_r = int(s_box[random_numbers[i][j][0], random_numbers[i][j][1]], 16) ^ int(s_box[row_r, column_r], 16)
            new_g = int(s_box[random_numbers[i][j][2], random_numbers[i][j][3]], 16) ^ int(s_box[row_g, column_g], 16)
            new_b = int(s_box[random_numbers[i][j][4], random_numbers[i][j][5]], 16) ^ int(s_box[row_b, column_b], 16)

            im[i, j] = new_b, new_g, new_r

    result_queue.put((im, image_id))
```

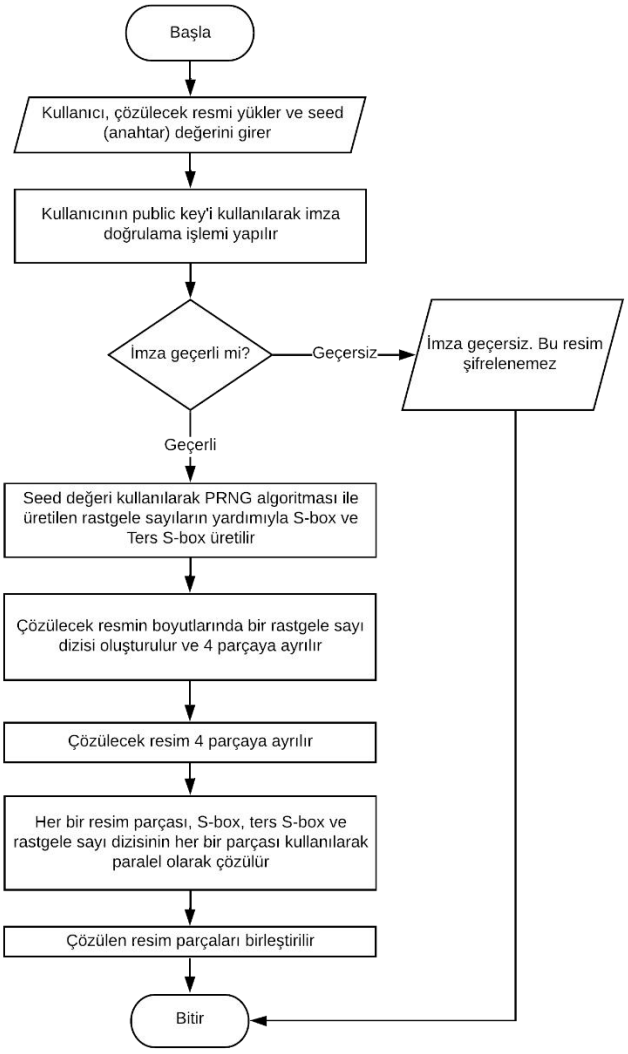
*Şekil 2.6. Şifreleme Algoritmasının Çalıştırıldığı Metot*

## 2.3. Çözme Aşaması




*Şekil 2.7. Çözme Aşaması*

Şekil 2.7'de görüldüğü üzere, kullanıcının imzası, RSA dijital imza algoritması ile kullanıcının public key'i kullanılarak doğrulanır. İmzanın geçerli olması halinde imzalanmış şifreli resim 4 parçaya ayrılır. Her bir parça paralel olarak simetrik şifreleme algoritmasına sokulup, kullanıcının girdiği seed (anahtar) değeri kullanılarak çözülür. Çözülmüş parçalar birleştirilir ve orijinal resim elde edilir. Çözme aşamasının akış diyagramı aşağıdaki gibidir:




*Şekil 2.8. Çözme Aşamasının Akış Diyagramı*

Şekil 2.8'de anlatıldığı üzere, resmi çözecek olan kullanıcı, çözülecek resmi yükler ve belirlediği seed (anahtar) değerini girer. Kullanıcının imzası, RSA dijital imza algoritması ile kullanıcının public key'i kullanılarak doğrulanır. İmza geçersiz veya başka birine ait ise resim şifrelenemez ve işlem son bulur. İmzanın geçerli olması halinde, girilen seed değerine göre PRNG (Pseudo-Random Number Generator) algoritması kullanılarak rastgele sayı üretimi başlar. Oluşturulan rastgele sayılar kullanılarak S-box matrisinde bulunan sayılar karıştırılır ve S-box üretimi sonlandırılır. S-box matrisi kullanılarak ters S-box matrisi oluşturulur. Çözülecek resmin boyutlarında bir rastgele sayı dizisi oluşturulur ve 4 parçaya ayrılır. Ayrıca çözülecek

resim de 4 parçaya ayrılır. Daha sonra multiprocessing yöntemi kullanılarak 4 ayrı process oluşturulur. Bu process'lere her bir resim parçası, S-box, ters S-box ve rastgele sayı dizisinin her bir parçası sokulur. Ardından bütün parçalar simetrik şifreleme algoritması kullanılarak paralel olarak çözülür. Çözülen parçalar birleştirilir ve orijinal resim elde edilir.

```
def decrypt(obj, password):
    en_img = cv2.imread(obj.image.path)

    height = int(len(en_img))
    width = int(len(en_img[0]))

    array_slicer = Slicer(en_img, height, width)
    en_img_top_left, en_img_top_right, en_img_bottom_left, en_img_bottom_right = array_slicer.slice()

    np.random.seed(int(password))

    shuffle = KnuthShuffle()
    s_box = shuffle.create_s_box(np.random)
    inverse_s_box = shuffle.create_inverse_s_box()

    random_numbers = np.random.randint(0, 16, (height, width, 6))
    array_slicer.set_array(random_numbers)
    rand_top_left, rand_top_right, rand_bottom_left, rand_bottom_right = array_slicer.slice()

    image_encryption = ImageEncryption()

    start = time.perf_counter()

    result_queue = Queue()

    procs = []
    proc1 = Process(target=image_encryption.decrypt,
                    args=(s_box, inverse_s_box, rand_top_left, en_img_top_left, result_queue, 1))
    procs.append(proc1)
    proc1.start()

    proc2 = Process(target=image_encryption.decrypt,
                    args=(s_box, inverse_s_box, rand_top_right, en_img_top_right, result_queue, 2))
    procs.append(proc2)
    proc2.start()

    proc3 = Process(target=image_encryption.decrypt,
                    args=(s_box, inverse_s_box, rand_bottom_left, en_img_bottom_left, result_queue, 3))
    procs.append(proc3)
    proc3.start()

    proc4 = Process(target=image_encryption.decrypt,
                    args=(s_box, inverse_s_box, rand_bottom_right, en_img_bottom_right, result_queue, 4))
    procs.append(proc4)
    proc4.start()

    image_slice_list = [result_queue.get() for i in range(4)]
    image_slice_list.sort(key=Util.sort_second)

    for proc in procs:
        proc.join()

    finish = time.perf_counter()

    print('Finished in {} second(s)'.format(finish - start))

    decrypted_image = Slicer.concatenate(image_slice_list[0][0], image_slice_list[1][0], image_slice_list[2][0],
                                         image_slice_list[3][0])

    media_root = settings.MEDIA_ROOT
    write_path = os.path.join(settings.MEDIA_ROOT, 'decrypted/'+obj.name+str(obj.id)+'.png')
    cv2.imwrite(write_path, decrypted_image)

    return 'https://imsafe.systems'+settings.MEDIA_URL+'decrypted/'+obj.name+str(obj.id)+'.png'
```

*Şekil 2.9. Process'lerin Oluşturulduğu Çözme Metodu*

Şekil 2.9'da process'lerin oluşturulduğu çözme metodu görülmektedir. Bu metot ayrıca resim parçalama, S-box oluşturma, ters S-box oluşturma, resim parçalarını birleştirme ve Şekil 2.10'da görülen çözme algoritmasının çalıştırıldığı metotları çağırır.

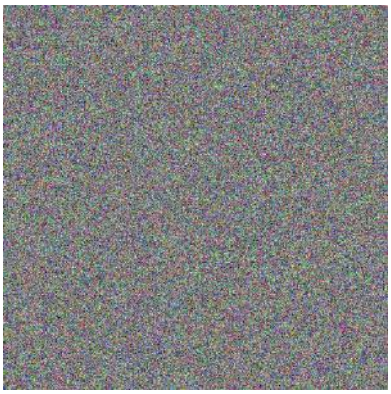
```
def decrypt(self, s_box, inverse_s_box, random_numbers, im, result_queue, image_id):
    for i in range(len(im)):
        for j in range(len(im[0])):
            b, g, r = im[i][j]

            hex_r = Util.convert_dec_to_hex(r)
            hex_g = Util.convert_dec_to_hex(g)
            hex_b = Util.convert_dec_to_hex(b)

            new_r = int(s_box[random_numbers[i][j][0], random_numbers[i][j][1]], 16) ^ int(hex_r, 16)
            new_g = int(s_box[random_numbers[i][j][2], random_numbers[i][j][3]], 16) ^ int(hex_g, 16)
            new_b = int(s_box[random_numbers[i][j][4], random_numbers[i][j][5]], 16) ^ int(hex_b, 16)

            new_r = Util.convert_dec_to_hex(new_r)
            new_g = Util.convert_dec_to_hex(new_g)
            new_b = Util.convert_dec_to_hex(new_b)

            im[i, j] = int(inverse_s_box[int(new_b[0], 16), int(new_b[1], 16)], 16), int(
                inverse_s_box[int(new_g[0], 16), int(new_g[1], 16)], 16), int(
                inverse_s_box[int(new_r[0], 16), int(new_r[1], 16)],
                16)

    result_queue.put((im, image_id))
```

*Şekil 2.10. Çözme Algoritmasının Çalıştırıldığı Metot*

## 2.4. Yapılan İşlemlerin Sonucu

İşlemler sonucunda elde edilen şifrelenmiş ve çözülmüş resimler aşağıdaki gibidir:

*Şekil 2.11. Orijinal Resim*



*Şekil 2.12. Şifrelenmiş Resim*

*Şekil 2.13. Çözülmüş Resim*

## 2.5. İşlemlerin Paralel Olarak Çalışmasının Etkisi

Test amacıyla 1024x768 boyutlarında bir resim şifrelenip çözüldü. Sıralı olarak çalışan işlemlerin süreleri aşağıdaki gibidir:

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.19
>>> runfile('C:/Users/DELL/Desktop/GitHub Repositories/
Encryption finished in 32.7663169 second(s)
```

*Şekil 2.14. Sıralı Olarak Çalışan Şifreleme İşleminin Süresi*

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.
>>> runfile('C:/Users/DELL/Desktop/GitHub Repositorie
Decryption finished in 35.7235323 second(s)
Structural Similarity Index: 1.0
```

*Şekil 2.15. Sıralı Olarak Çalışan Çözme İşleminin Süresi*

Paralel olarak çalışan işlemlerin süreleri aşağıdaki gibidir:

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [M
>>> runfile('C:/Users/DELL/Desktop/GitHub Reposi
Encryption finished in 11.1649332 second(s)
```

*Şekil 2.16. Paralel Olarak Çalışan Şifreleme İşleminin Süresi*

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13)
>>> runfile('C:/Users/DELL/Desktop/GitHub Repo
Decryption finished in 11.7829477 second(s)
Structural Similarity Index: 1.0
```

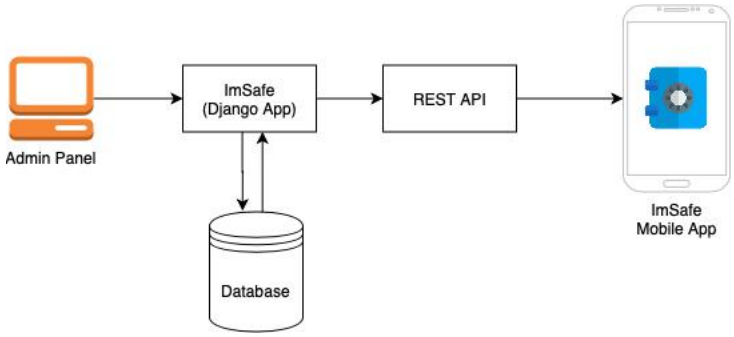
*Şekil 2.17. Paralel Olarak Çalışan Çözme İşleminin Süresi*

Yapılan gözlemler sonucunda, 1024x768 boyutunda bir resim için işlem sürelerinin yaklaşık olarak %65 oranında azaldığı tespit edildi. Buna ek olarak orijinal resim ile çözülmüş resmin yapısal benzerlik indeksi (Structural Similarity Index) hesaplanarak indeks değerinin 1.0 olduğu gözlendi. Bu değer, orijinal resim ile çözülmüş resmin birebir aynı olduğunu göstermektedir [3].

## 3. DJANGO RESTful Web Servis

### 3.1 Django RESTful Web Servisinin Uygulanması

Projenin sunucusu ve mobil uygulaması arasındaki iletişimini sağlamak amacıyla bir web servise ihtiyaç duyulmaktadır. Günümüzde oldukça sık kullanılan ve verimli bir yöntem olan RESTful web servis mimarisi tercih edilmiştir. Uygulama geliştirme adımlarını hızlandırmak ve geliştirilen resim şifreleme yöntemini kullanıcılardan soyutlamak amacıyla Django REST Framework kullanılmıştır [4]. Sunucu üzerinde projede tanımlanan algoritmalar ve mobil uygulama ile iletişime geçecek olan servis arayüzü katmanlara ayrılmıştır.



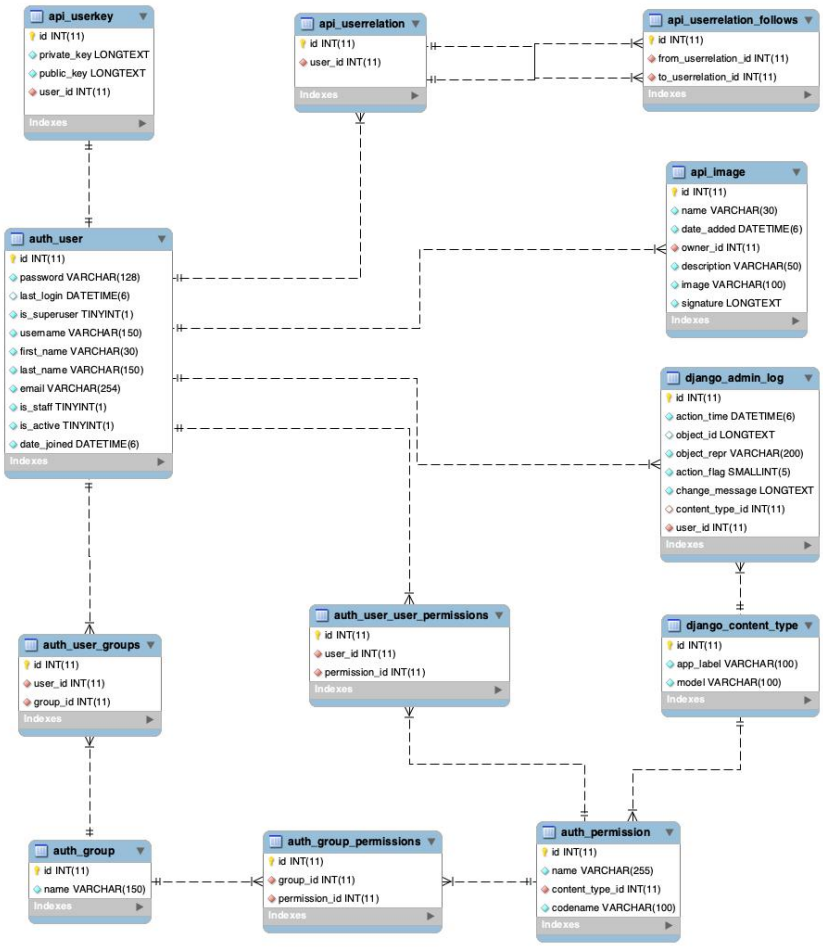

*Şekil 3.1. Web Servis Veri Akış Modeli*

Projenin iletişim yapısı Şekil 3.1'de verilmiştir. Sunucuda Django kullanılarak oluşturulmuş olan ImSafe uygulaması çalışmaktadır. Uygulamanın, veri tabanı ve bir üst katmanda çalışmakta olan REST API arasında veri iletişimi bulunmaktadır. Mobil uygulama yalnızca REST API ile bağlantı durumundadır. Bu şekilde soyutlama işlemi yapılarak standart

kullanıcıların erişebileceği tek arayüz API olarak belirlenmiştir. Yönetici yetkisine sahip olan kullanıcılar ise direkt olarak sunucuda bulunan ImSafe servislerine erişebilmektedir.

Sunucuda bulunan uygulama MVC (Model – View – Controller) mimarisi kullanılarak geliştirilmiştir. MVC mimarisi uygulamanın karmaşıklığını azaltmakta ayrıca REST arayüzü ve mobil uygulama ile ortak bir yapı oluşturarak ölçeklenebilir, geliştirmesi, yönetmesi daha kolay bir ortam sunmaktadır. API arayüzü ile mobil uygulama arasındaki veri akışı JSON veri tipi ile kodlanmıştır. Sunucuda çalışan ImSafe uygulaması model katmanındaki objelerin JSON türüne dönüştürülmesi için Model Serializer nesnelerini kullanmaktadır. API üzerinden gelen ve giden veriler her modele özel Serializer objesi ile verilerin geçerliliğini (Data Validation) kontrol etmektedir. Verilerin uygunluk testi başarılı olduğu durumda API'dan mobil uygulamaya giden veriler JSON nesnesine, mobil uygulamadan gelen veriler ise sunucuda işlenmek üzere Django nesnelerine dönüştürülmektedir.

## 3.2 Varlık-İlişki Modeli

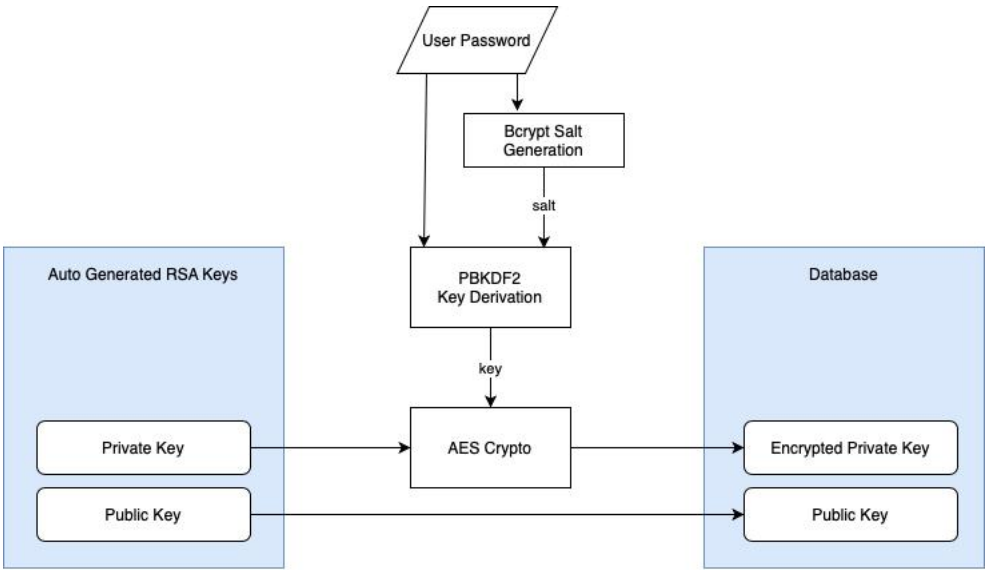



*Şekil 3.2. Varlık-İlişki Modeli*

Sunucu uygulamasının nesneler arası ilişki diyagramı Şekil 3.2'de verilmiştir. Web servisinin geliştirilme aşamasında kullanılan Django Framework'ün sağlamış olduğu kullanıcı, kullanıcı grubu, kullanıcı izinleri ve işlem kayıt nesneleri ihtiyaca göre genişletilerek kullanılmıştır. Kullanıcıların RSA anahtarlarını kaydetmek amacıyla "userkey", şifrelenerek depolanacak resimler için "image" ve kullanıcıların takipleşme özelliği sayesinde birbirlerine resim transfer edebilmeleri amacıyla "userrelation" tabloları oluşturulmuştur. Bu tablolar ile oluşturulan model projenin temel işlevlerini yerine getirecek niteliktedir.

Her kullanıcıya ait bir adet "userkey" nesnesi bulunmaktadır. RSA anahtarları kullanıcı özelinde tek bir nesne üzerinden gerçekleşmektedir. Bir kullanıcı birden fazla resim yükleyebilir ve transfer edebilir. Kullanıcı takip etme özelliği için oluşturulan tabloda tek yönlü bir dizi tutulmaktadır. Bir kullanıcı takip ettiği kişileri ve onu takip eden kişileri listeleyebilir.

### 3.3 RSA Anahtarlarının Şifrelenmesi




*Şekil 3.3. RSA Anahtarının AES ile Şifrelenme Modeli*

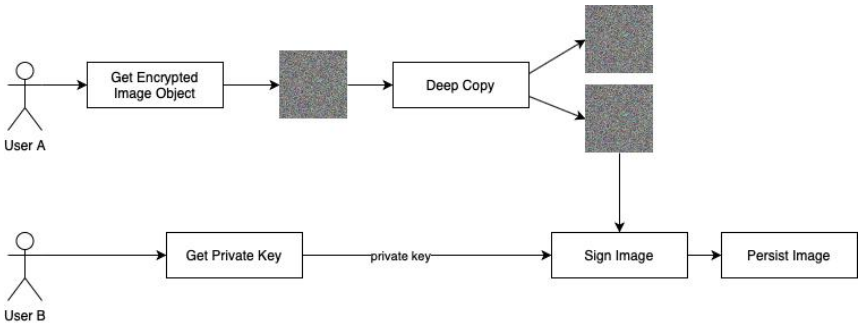
Sisteme yüklenen resimlerin imzalanma aşamasında kullanılan RSA anahtarları, kullanıcılardan soyutlanarak uygulamanın kullanımını kolaylaştırması amaçlanmıştır. Gizli anahtarların şifresiz metin olarak saklanması güvenlik sorunları oluşturacağı için gizli anahtarları saklamak amacıyla Şekil 3.3'te verilen model kullanılmıştır. Anahtarların saklanmasında en güvenli yöntemlerden biri fiziksel cihazlarda şifrelenerek tutulmasıdır ancak uygulamanın geliştirilme aşamasında yazılımsal olarak fiziksel cihaz simule edilerek benzer bir yapı kurulmuştur.

RSA anahtarları simetrik bir algoritma olan AES ile şifrelenmektedir [5]. AES algoritmasında key olarak kullanılmak üzere kullanıcının uygulamaya üye olurken seçmiş olduğu şifreden yararlanılmaktadır. Üye olma işleminin ilk aşamasında otomatik olarak kullanıcıya özel RSA anahtarları üretilmektedir. Kullanıcının belirlemiş olduğu şifre Bcrypt algoritması kullanılarak benzersiz bir salt değeri üretilmektedir [6]. Üretilen salt değeri ve kullanıcının şifresi PBKDF anahtar üretim algoritmasına girdi olarak verilmektedir [7]. Bu işlemler sonucunda kullanıcıya özel sabit uzunlukta benzersiz bir anahtar elde edilmektedir. Üretilen anahtar AES algoritmasının key değeri, kullanıcının RSA gizli anahtarı ise plaintext değeri olarak girdi olmaktadır. AES algoritması sonucunda oluşmakta olan şifrelenmiş anahtar veri tabanında tutulmaktadır. İhtiyaç halinde veri tabanında bulunan şifreli anahtar çözülerek kullanılmaktadır. Kullanıcı şifresi değişmesi halinde gizli anahtar deşifre edilmekte ve kullanıcının yeni şifresi kullanılarak tekrar aynı adımlar izlenerek veri tabanına kaydedilmektedir.


*Şekil 3.4. RSA Anahtarlarının Depolanma Yapısı*

Şekil 3.4'te kullanıcının şifrelenmiş olan gizli anahtarı ve açık anahtarları gösterilmektedir. Kullanıcı anahtarları Base64 algoritması ile string veri tipine dönüştürülerek saklanmaktadır. Anahtarların kullanılacağı işlemlerde byte dizisine dönüştürülerek ilgili algoritmalara girdi olarak verilmektedir.

## 3.4 Resim Transfer İşlemi




*Şekil 3.5. Transfer İşleminin Aşamaları*

Kullanıcılar arası resim transfer işlemi şifrelenmiş resimler ile gerçekleştirilmektedir. Resim transfer etmek isteyen A kullanıcısı, B kullanıcısına göndermek istediği resmi seçtiğinde sunucuda bulunan şifreli resim kopyalanmaktadır. B kullanıcısının gizli anahtarı veri tabanından okunarak kopyalanan resmi imzalama aşamasında kullanılmaktadır. İmzalama işlemi sonucunda aynı resmin A ve B kullanıcıları için ayrı ayrı imzalanmış objeleri elde edilmektedir. B kullanıcısı adına imzalanan obje veri tabanına yazılarak transfer işlemi tamamlanmaktadır. Transfer işlemi sonucunda B kullanıcısı sahip olduğu resimler arasında A kullanıcısının göndermiş olduğu resmi listeleyebilmektedir. Resim transfer işleminde derin kopyalama adımı sayesinde A kullanıcısı göndermiş olduğu resmi kendi listesinden silmesi durumunda B kullanıcısı resme erişimi devam etmektedir.

### 3.5 API Fonksiyonları

Mobil uygulama ve sunucu arasındaki iletişimi sağlamak üzere hazırlanan API arayüzünün bağlantı uçları sunucu uygulamanın fonksiyonlarını çalıştırmaktadır. HTTP durum kodlarına uygun olarak hazırlanan fonksiyonlar mobil uygulama kullanıcısının isteklerine cevap vermektedir. Kimlik doğrulama, yetkilendirme ve veri doğrulama işlevleri API üzerinde tanımlanmıştır. Oluşabilecek hatalar durum kodlarıyla kontrol edilmekte, sunucuya gelen ve giden bağlantılar kayıt altına alınmaktadır. Log kayıtları yalnızca sunucu güvenliğini sağlamak amacıyla tutulmakta olup kullanıcıların kişisel bilgileri bulunmamaktadır.

Kullanıcı doğrulama işlemi en temel doğrulama yöntemlerinden olan kullanıcı adı ve şifresi kullanılarak yapılan Basic Authentication ile gerçekleşmektedir. Sunucu uygulamanın modeli ölçeklenebilir bir yapıda olmasından dolayı Token Authentication gibi sık kullanılan yöntemleri tanımlamak için uygun bir yapıdadır. Yetkilendirme işlemini yönetici yetkisine sahip kullanıcılar Admin Panel üzerinden gerçekleştirebilmektedir. Mobil uygulama üzerinden kayıt olan kullanıcılar standart kullanıcı rolünde tanımlanmaktadır.

API bağlantı uç noktaları (endpoint) fonksiyona özel durumlar haricinde GET, POST, PUT, PATCH ve DELETE metotlarını çalıştıracak şekilde tanımlanmıştır. REST Framework üzerinde tanımlanmış olan fonksiyonlar aşağıdaki gibidir. Her bir fonksiyon Test ve Production Server üzerinde test edilerek veri alışverişi kontrol edilmiştir. Fonksiyon testleri Postman uygulaması ile yapılmıştır.

- /api/users [GET] [POST]
  Kullanıcıların listelenmesi ve yeni kullanıcı eklemek için kullanılır.

- /api/users/{user_id} [GET] [PUT] [PATCH] [DELETE]

  ID numarası verilen kullanıcı bilgilerini listelemek, düzenlemek ve silmek için kullanılır.
- /api/followings [GET]

  Takip edilen kullanıcıları listelemek için kullanılır.
- /api/followers [GET]

  Takip eden kullanıcıları listelemek için kullanılır.
- /api/follow [POST]

  Kullanıcı takip etmek için kullanılır.
- /api/unfollow [POST]

  Kullanıcı takip etmeyi bırakmak için kullanılır.
- /api/search-users [POST]

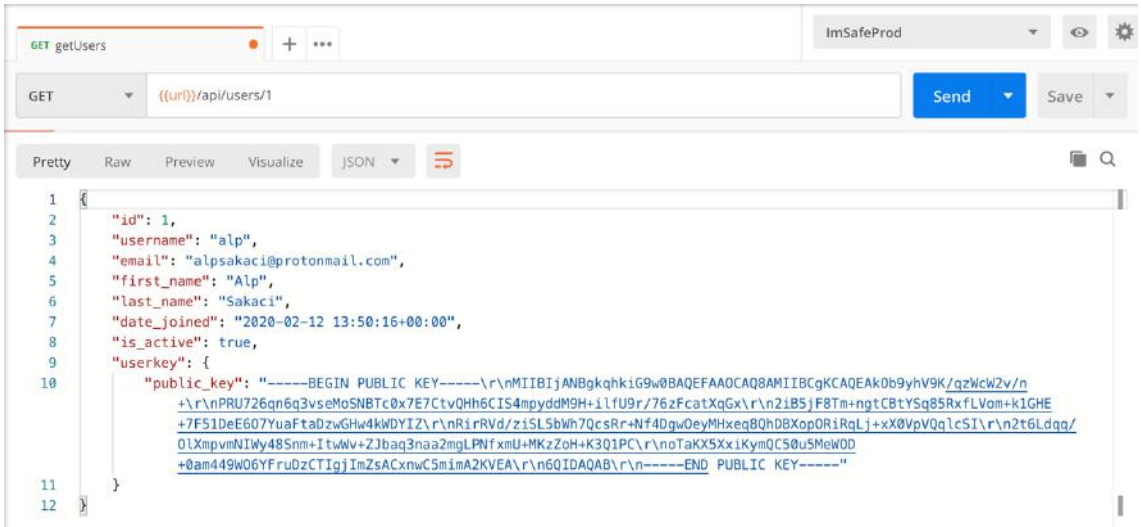
  Kullanıcı arama işlemi için kullanılır.
- /api/images [GET] [POST]

  Kimliği doğrulanmış olan kullanıcıya ait olan resim bilgilerini listelemek ve yeni resim eklemek için kullanılır. Eklenen resim şifrelenir.
- /api/images/{image_id} [GET] [PUT] [PATCH] [DELETE]

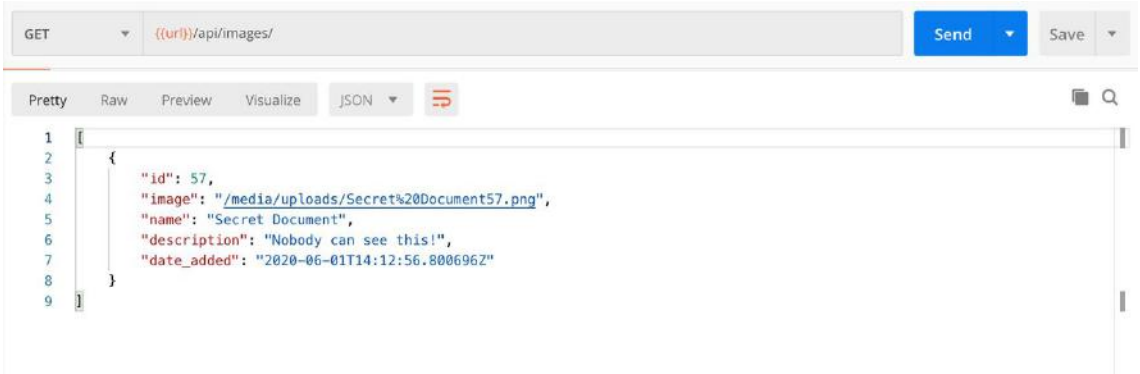
  ID numarası verilen resim bilgileri listelemek, düzenlemek ve silmek için kullanılır.
- /api/images/{image_id}/decrypt [POST]

  ID numarası verilen resmi deşifreler.
- /api/images/{image_id}/transfer [POST]

  ID numarası verilen resmi başka bir kullanıcıya transfer etmek için kullanılır.



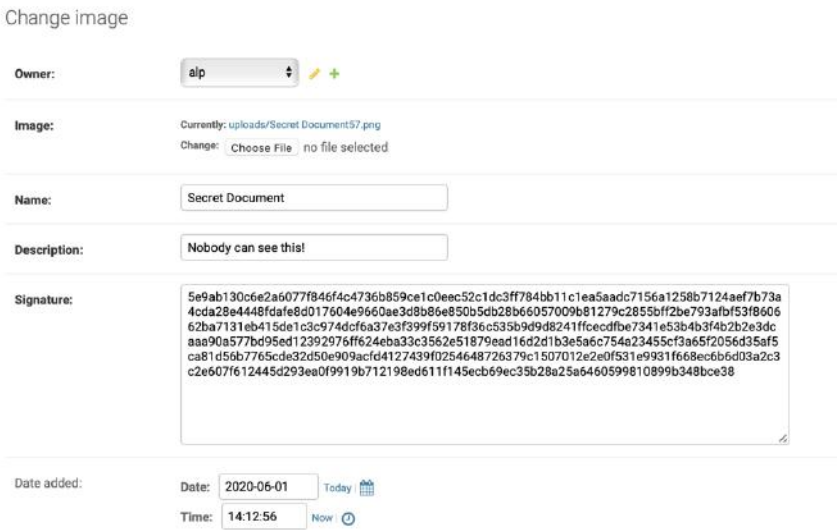

*Şekil 3.6. API Üzerinden Alınan Kullanıcı Bilgisi Çıktısı*



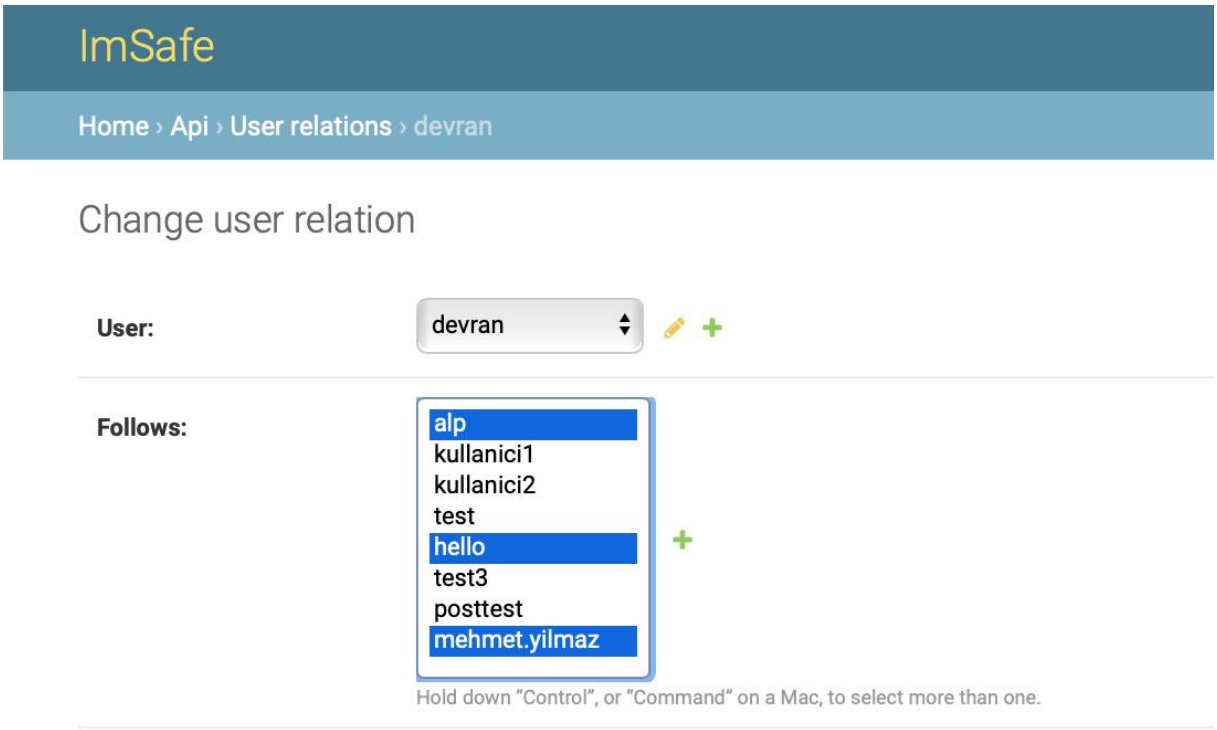

*Şekil 3.7. API Üzerinden Alınan Resim Bilgisi Listeleme Çıktısı*

## 3.6 Admin Paneli

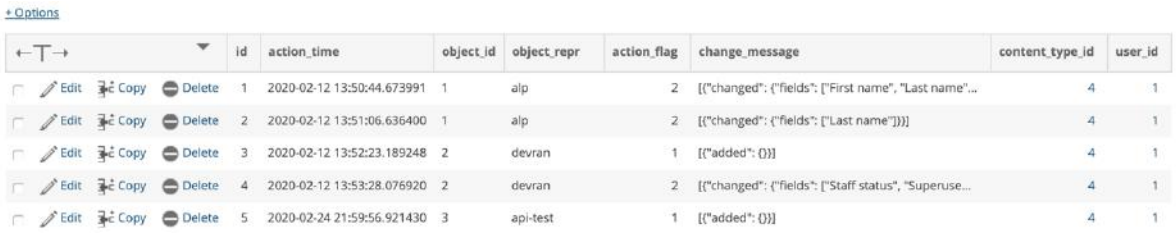
Yönetici yetkisine sahip olan kullanıcıların web arayüzünden ImSafe servislerine erişebilmesi için REST API'dan bağımsız bir Admin Paneli hazırlanmıştır. Admin panelinin hazırlanma aşamasında Django Framework'ün sunmuş olduğu eklentilerden yararlanılmıştır. Projede tanımlanan model ve metotlara erişim sağlanmıştır. Panel üzerinden kullanıcı bilgileri ve kullanıcı grupları, yetkiler, kullanıcıların takip listesi, bulut üzerindeki resim bilgileri görüntülenebilmektedir. Yönetici kullanıcılar yalnızca yetkileri dahilinde görüntüleme ve değişiklik yapabilmektedir. Yöneticilerin yapmış olduğu değişiklikler kayıt altına alınmaktadır. Veri tabanına eklenen kayıtlar sistemin güvenliğinin sağlanmasına ve yapılan işlemlerde oluşabilecek hataların incelenmesine yardımcı olmaktadır. Kullanıcıların bulut hizmetine yüklemiş olduğu resimler şifreli olduğu için sisteme giriş yapan yöneticiler dahi resimlerin içeriğini görüntüleyememektedir.




*Şekil 3.8. Admin Paneli Üzerinden Resim Bilgilerinin Görüntülenmesi*




*Şekil 3.9. Admin Paneli Üzerinden Takip Edilen Kullanıcıların Görüntülenmesi*

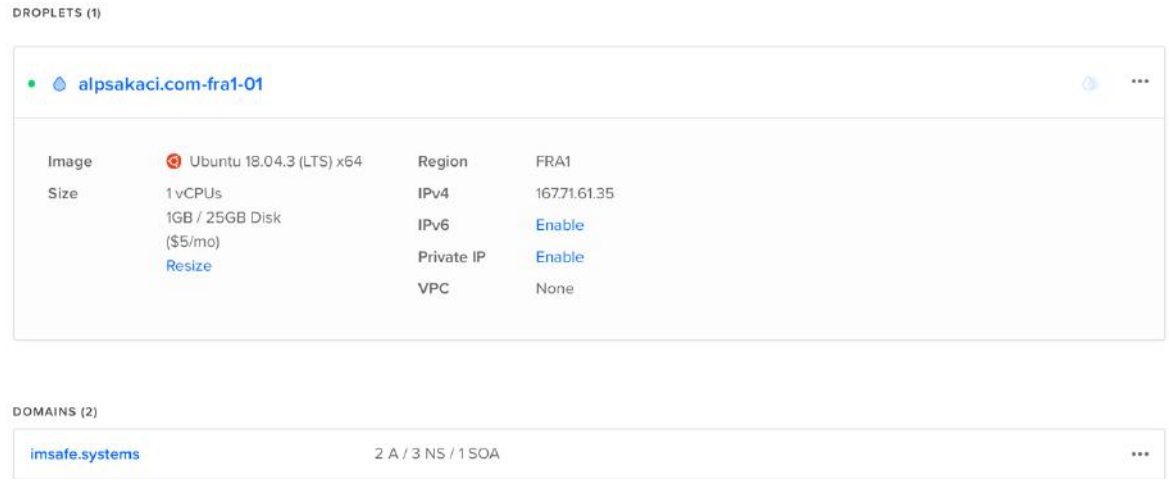



*Şekil 3.10. Yapılan İşlemlerin Kayıtları*

### 3.7 Uygulamanın Bulut Sunucuya Kurulması

ImSafe uygulaması yerel sunucuda test edilerek tanımlanan fonksiyonların işlevlerini yerine getirdiği gözlemlenmiştir. Uygulamanın bulut servisi olarak çalışabilmesi için gerekli olan sunucu DigitalOcean isimli bulut altyapı sağlayıcı firmadan kiralanmıştır [8]. Kiralanan sunucuya ilk olarak Ubuntu işletim sistemi kurulmuştur. İşletim sistemi kurulumu ardından sunucunun dış dünyaya açılabilmesi için Apache Web Server kurularak firewall ve network ayarlamaları yapılmıştır. GitHub'da bulunan kaynak kodları sunucuya aktarılarak Apache üzerinde çalışması için konfigürasyon dosyaları hazırlanmıştır. MySQL veritabanı ile bağlantısı tamamlanan uygulama bulut servisi olarak kullanıma hazır hale getirilmiştir.

Sunucunun güvenli bağlantı sunabilmesi için SSL sertifikası alınarak Apache üzerine eklenmiştir [9]. Web sunucusu yalnızca HTTPS (443) portu üzerinden hizmet vermekte olup HTTP (80) portuna gelen bütün istekler 443 portuna yönlendirilmiştir. REST API üzerinden yapılan tüm veri alışverişi şifrelenmiş olup verileri yalnızca istemci ve sunucu işleyebilmektedir. Trafiği dinleyen kişiler olsa dahi kullanıcıların bilgileri ve bulut deposuna yüklenen resimleri görüntülenememektedir. Sunucu bağlantısı Fiddler yazılımı ile dinleme yapılarak test edilmiştir. Tüm bağlantıların şifrelenmiş ve içeriği gizli olduğu gözlemlenmiştir. Çevrimiçi araçlar kullanılarak bulut servisinin SSL sertifikası incelenmiştir. SSL rapor sonucunda sertifikanın yeterli bir puan aldığı gözlemlenmiştir.




*Şekil 3.11. Kiralanan Bulut Sunucu Bilgileri*


*Şekil 3.12. Fiddler Yazılımı ile SSL Bilgilerinin Listelenmesi*

# 4. ANDROID MOBİL UYGULAMASI

## 4.1. Android Uygulamasında Django RESTful Web Servisin Çağırılması

Web servisin çağırılmasında Retrofit kütüphanesi kullanılırken, HTTP istemcisinin oluşturulmasında OkHttp kütüphanesi kullanıldı.

```
12  public class ServiceGenerator {
13      private static final String API_BASE_URL = "https://imsafe.systems/api/";
14
15      private static OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
16
17      private static Retrofit.Builder builder =
18              new Retrofit.Builder()
19                      .baseUrl(API_BASE_URL)
20                      .addConverterFactory(GsonConverterFactory.create());
21
22      private static Retrofit retrofit = builder.build();
```

*Şekil 4.1. Web Servisin Çağırılması*

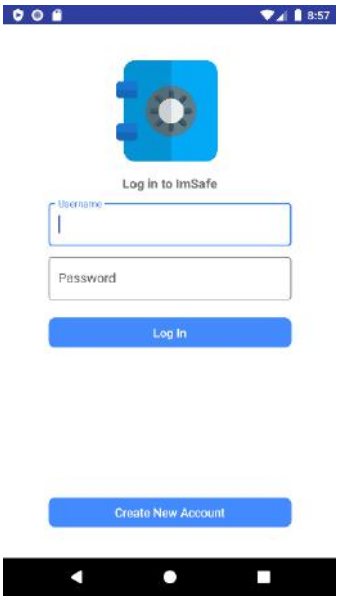
Şekil 4.1'de görüldüğü üzere geliştirilen web servisin base URL'i 'API_BASE_URL' adlı değişkene atandı. Daha sonra bir HTTP istemcisi oluşturuldu.

```
21  public interface ImSafeService {
22
23      @GET("images/")
24      Call<List<Image>> getImageList();
25
26      @Multipart
27      @POST("images/")
28      Call<ImageEncryptionResponse> encryptImage(@Part MultipartBody.Part image,
29                                                 @Part("name") RequestBody name,
30                                                 @Part("description") RequestBody description,
31                                                 @Part("password") RequestBody password);
32
33      @Multipart
34      @POST("images/{id}/decrypt/")
35      Call<ImageDecryptionResponse> decryptImage(@Path("id") String id, @Part("password") RequestBody password);
36
37      @Multipart
38      @POST("users/")
39      Call<ResponseBody> createAccount(@Part("username") RequestBody username,
40                                       @Part("email") RequestBody email,
41                                       @Part("password") RequestBody password,
42                                       @Part("first_name") RequestBody firstName,
43                                       @Part("last_name") RequestBody lastName);
```

*Şekil 4.2. API Endpoint'lerinin Kullanılması*

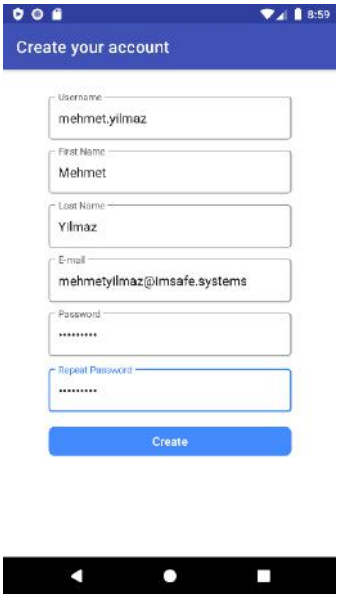
Şekil 4.2'de görüldüğü üzere API endpoint'lerine erişim sağlandı. Bu şekilde, örnek olarak şifreli resim listesini çekme, resim şifreleme, resim çözme ve hesap oluşturma endpoint'leri gösterilmiştir.

## 4.2. Android Uygulamasının Kullanımı




*Şekil 4.3. Kullanıcı Girişi Ekranı*

Şekil 4.3'te görüldüğü üzere uygulama ilk yüklendiğinde, kullanıcıyı giriş ekranı karşılamaktadır.




*Şekil 4.4. Kayıt Olma Ekranı*

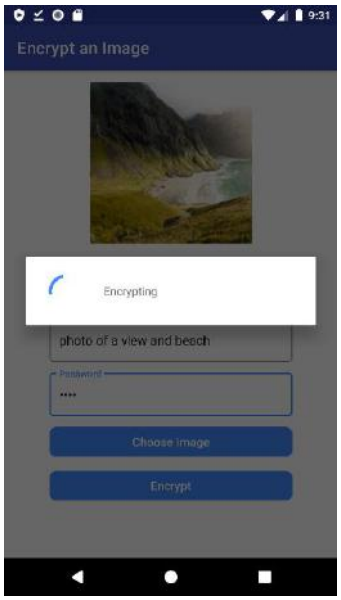

*Şekil 4.5. Kullanıcı Girişi Ekranı*

Uygulamanın kullanımını göstermek amacıyla 'Mehmet Yılmaz' adında bir kullanıcı oluşturuldu. Kullanıcı uygulamaya kayıt olmak için Şekil 4.4'teki ekrana kullanıcı adını, adını, soyadını, e-mail adresini ve şifresini girer. Daha sonra Şekil 4.5'te görüldüğü üzere giriş ekranında kullanıcı adını ve şifresini girer ve uygulamaya giriş yapar.

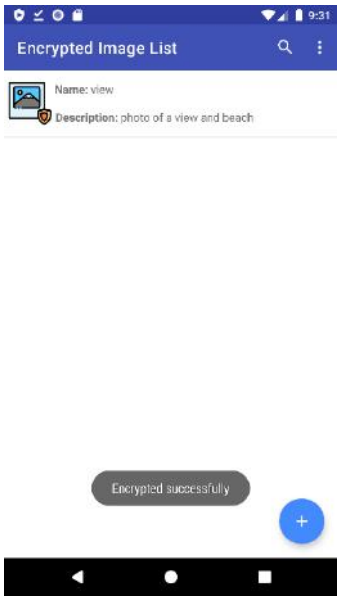

Şekil 4.6. Şifreli Resim Listesi Ekranı


Şekil 4.7. Resim Şifreleme Ekranı

Şekil 4.6'da görüldüğü üzere, kullanıcı, uygulamaya giriş yaptıktan sonra boş bir şifreli resim listesiyle karşılaşır. Resim şifrelemek için '+' butonuna basar ve galeriye yönlendirilir. Kullanıcı, galeriden şifrelemek istediği resmi seçtikten sonra Şekil 4.7'deki resim şifreleme ekranına yönlendirilir. Bu ekranda, resmin ismini, açıklamasını ve anahtarını (şifresini) girdikten sonra 'Encrypt' butonuna basar.

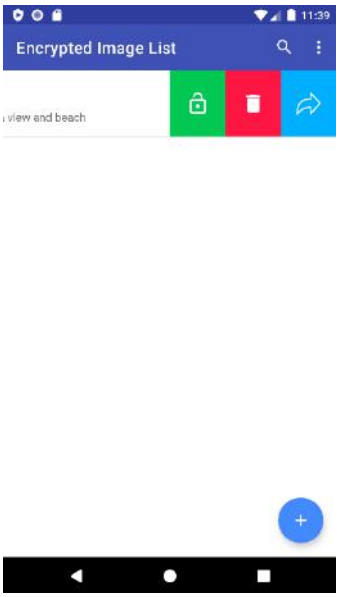



Şekil 4.8. Şifrelenme Aşaması

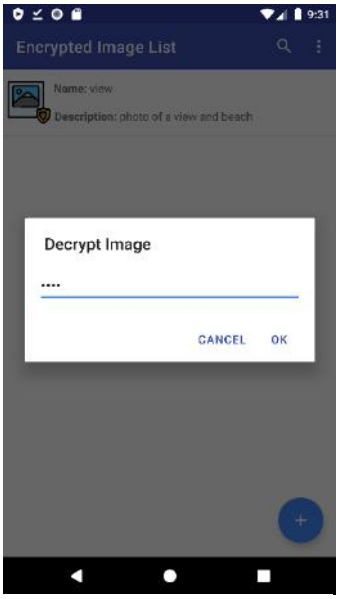



Şekil 4.9. Şifreli Resim Listesi

Şekil 4.9'da görüldüğü üzere, kullanıcı, resmin şifrelenme aşaması bittikten sonra şifreli resim listesine yönlendirilir.

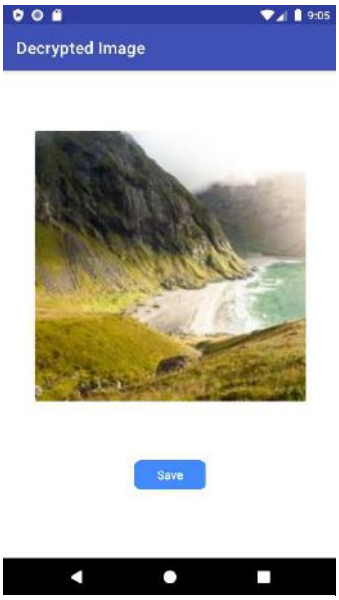



*Şekil 4.10. Kaydırmalı Menü*

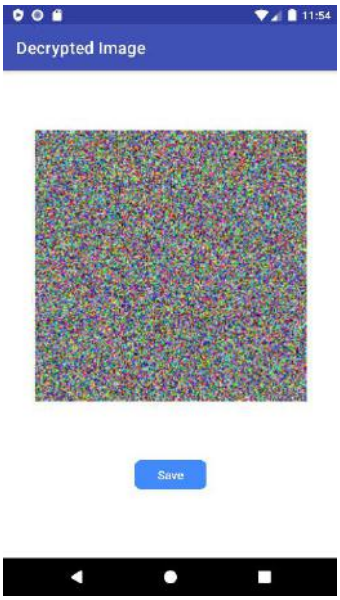



*Şekil 4.11. Resim Çözme Ekranı*

Resmi çözmek isteyen kullanıcı, çözmek istediği resmi sola kaydırır ve yeşil renkli açık kilit simgesine sahip butona basar. Karşısına resmin şifresini girmesi için bir popup gelir ve buraya şifreyi yazar ve 'OK' butonuna basar.

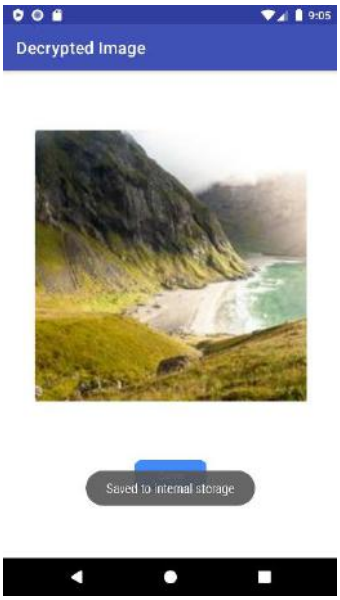



*Şekil 4.12. Çözülmüş Resim Ekranı*

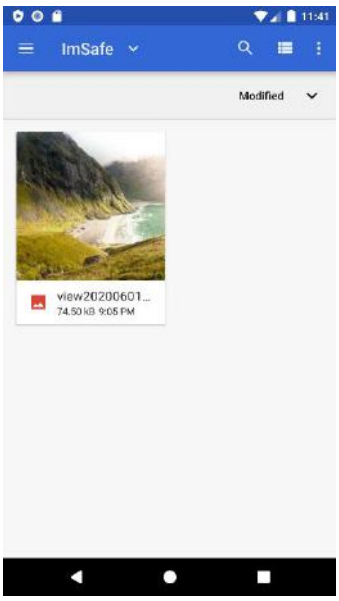



*Şekil 4.13. Yanlış Çözülen Resim*

Şekil 4.12'de görüldüğü üzere şifre doğru girildiğinde şifreli resim orijinal haline geri döndürülür. Şifre yanlış girildiği takdirde ise Şekil 4.13'teki gibi orijinal haline döndürülemez ve anlamsız bir resim ile karşılaşılır.



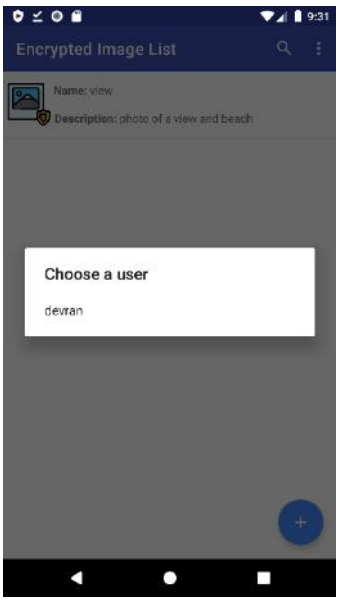

Şekil 4.14. Çözülmüş Resmin Galeriye Kaydedilmesi




Şekil 4.15. Galeri Ekranı

Çözülmüş resmi galerisine kaydetmek isteyen kullanıcı ‘Save’ butonuna basar. Ardından resim, galeriye ‘ImSafe’ klasörünün altına kaydedilir.

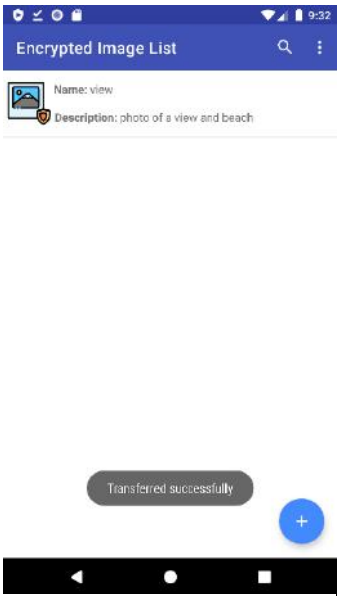

Şekil 4.16. Arama ve Takip Etme Ekranı




Şekil 4.17. Resim Transfer Ekranı

Şifrelediği resmi transfer etmek isteyen kullanıcının, resmi göndereceği kullanıcıyı takip etmesi gerekmektedir. Kullanıcı, şifreli resimlerin bulunduğu ekranda büyüteç ikonuna basar ve Şekil 4.16’da görülen kullanıcı arama ekranına yönlendirilir. Bu örnekte ‘devran’ kullanıcı adına sahip kullanıcıya resim transferi yapılacağı için bu kullanıcı takip edilmiştir. Ardından resim listesine döner ve transfer etmek istediği resmi sola kaydırır. Mavi renkli ok simgesine sahip

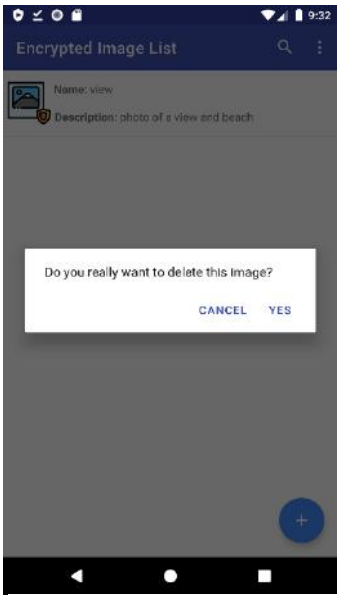
butona basar. Karşısına takip ettiği kullanıcı listesi gelir. Buradan ‘devran’ isimli kullanıcıyı seçer ve resim transfer işlemi sona erer.




*Şekil 4.18. Şifreli Resim Listesi*


*Şekil 4.19. 'devran' Kullanıcısının Şİfreli Resim Listesi*

Şekil 4.19’da ‘devran’ kullanıcı adına sahip kullanıcının şifreli resim listesi bulunmaktadır. Transfer edilen resmin bu kullanıcının şifreli resim listesine başarıyla eklendiği görülmektedir.




*Şekil 4.20. Resim Silme Ekranı*


*Şekil 3.21. Şifreli Resim Listesi*

Şifreli resim listesinden resim silmek isteyen kullanıcı, silmek istediği resmi sola kaydırır ve kırmızı renkli çöp kutusu simgesine sahip butona basar. Daha sonra karşısına onaylaması için bir popup çıkar. ‘YES’ butonuna bastığı takdirde silmek istediği resim başarıyla silinir.

## 5. SONUÇ

Bu çalışmada, Dr. Öğr. Üyesi Erdal Güvenoğlu'nun geliştirdiği simetrik resim şifreleme algoritmasına ek olarak paralel şifreleme ve RSA dijital imza algoritması ile imzalama işlemleri uygulanmıştır. Kullanıcının buluta yüklediği resimler bu işlemlere tabi tutulmuştur. Böylece depolanan resimler güvence altına alınmıştır. Bulut sunucu üzerinde çalışan bir web servis ve bu servisi kullanan bir mobil uygulama geliştirilmiştir.

## KAYNAKLAR

[1] Güvenoğlu, E. 2016. "Resim Şifreleme Amacıyla Dinamik S Kutusu Tasarımı İçin Bir Yöntem", El-Cezerî Fen ve Mühendislik Dergisi, Cilt: 3, No: 2, 2016 (179-191).

[2] Fisher-Yates shuffle, https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle.

[3] Structural similarity index (SSIM), Python Scikit-Image Documentation, https://scikit-image.org/docs/dev/auto_examples/transform/plot_ssim.html.

[4] Django Documentation, https://docs.djangoproject.com/en/3.0/.

[5] Pycrypto API Documentation, AES Module, https://www.dlitz.net/software/pycrypto/api/current/.

[6] Bcrypt Password Hashing Function https://pycryptodome.readthedocs.io/en/latest/src/protocol/kdf.html#bcrypt.

[7] PBKDF2 Key Derivation Algorithm https://pycryptodome.readthedocs.io/en/latest/src/protocol/kdf.html#pbkdf2.

[8] Digital Ocean, Create Droplets and Deploy Application, https://www.digitalocean.com/docs/droplets/quickstart/.

[9] Let's Encrypt, SSL Certificate https://letsencrypt.org.