BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİMDALI
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

GÖRÜNTÜ İLETİMİNDE WAVELET TABANLI RASTGELE PERMÜTASYON İLE BİR ŞİFRELEME METODU GELİŞTİRİLMESİ

HAZIRLAYAN

KENAN TUĞKAN DEMİRCİ

YÜKSEK LİSANS TEZİ

ANKARA - 2020

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİMDALI
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ TEZLİ YÜKSEK LİSANS PROGRAMI

GÖRÜNTÜ İLETİMİNDE WAVELET TABANLI RASTGELE PERMÜTASYON İLE BİR ŞİFRELEME METODU GELİŞTİRİLMESİ

HAZIRLAYAN

KENAN TUĞKAN DEMİRCİ

YÜKSEK LİSANS TEZİ

TEZ DANIŞMANI

Dr. Öğr. Üyesi AHMET GÜNGÖR PAKFİLİZ

ANKARA - 2020

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Elektrik-Elektronik Mühendisliği Anabilim Dalı Elektrik-Elektronik Mühendisliği Tezli Yüksek Lisans Programı çerçevesinde Kenan Tuğkan DEMİRCİ tarafından hazırlanan bu çalışma, aşağıdaki jüri tarafından Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 22 / 08 / 2020

Tez Adı: Görüntü İletiminde Wavelet Tabanlı Rastgele Permütasyon ile Bir Şifreleme Metodu Geliştirilmesi

Tez Jüri Üyeleri (Unvanı, Adı - Soyadı, Kurumu) **İmza**

Prof. Dr. Sıtkı Çağdaş İNAM – Başkent Üniversitesi ……..

Dr. Öğr. Üyesi Ahmet Güngör PAKFİLİZ – Başkent Üniversitesi ……..

Dr. Akın ÖZKAN – TÜBİTAK-SAGE ……..

ONAY

Prof. Dr. Ömer Faruk ELALDI

Fen Bilimleri Enstitüsü Müdürü

Tarih: … / … / ……..

BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: … / … / 20…

Öğrencinin Adı, Soyadı : Kenan Tuğkan DEMİRCİ

Öğrencinin Numarası : 21710295

Anabilim Dalı : Elektrik-Elektronik Mühendisliği

Programı : Elektrik-Elektronik Mühendisliği Tezli Yüksek Lisans

Danışmanın Unvanı/Adı, Soyadı : Dr. Öğr. Üyesi Ahmet Güngör PAKFİLİZ

Tez Başlığı : Görüntü İletiminde Wavelet Tabanlı Rastgele Permütasyon ile Bir Şifreleme Metodu Geliştirilmesi

Yukarıda başlığı belirtilen Yüksek Lisans tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam ....78.... sayfalık kısmına ilişkin, ….. / ..… / 20… tarihinde şahsım/tez danışmanım tarafından ……Turnitin…… adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % ….7….'dir.

Uygulanan filtrelemeler:

1. Kaynakça hariç

2. Alıntılar hariç

3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:…….

ONAY

… / … / 20…

Dr. Öğr. Üyesi Ahmet Güngör PAKFİLİZ

ÖZET

Kenan Tuğkan Demirci

GÖRÜNTÜ İLETİMİNDE WAVELET TABANLI RASTELE PERMÜTASYON İLE BİR ŞİFRELEME METODU GELİŞTİRİLMESİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

2020

Bu çalışmada; özellikle savunma sanayi alanında görüntü aktarımı esnasında, daha güvenli ~~iyi~~ bir şifreleme ile bilgi aktarımının vektör sağlanması hedeflenmiştir.

Bu çalışma sırasında farklı renk yoğunluklarına sahip dört farklı görüntü kullanılmıştır. Görüntüler sırasıyla frekans ve zaman uzayında şifrelenmiş, böylece anlamsız görüntüler oluşturularak aktarım esnasında güvenlikleri sağlanmıştır. Alıcıya veri aktarılırken, görüntü ile beraber iletilen bir sıralama vektörü sayesinde görüntünün anlaşılması güçleştirilerek güvenliği sağlanmıştır.

Vektör alıcıya iletilirken görüntü piksel sayısıyla ilişkili olarak LIFO (last in first out) veya FIFO (first in first out) algoritması kullanılarak verinin şifresinin başkaları tarafından çözülmesi güçleştirilmiştir.

Bu çalışmada, sadece zaman uzayında yapılan rastgele permütasyon işlemi ile, frekans uzayında gerçekleştirilmiş ayrık dalgacık dönüşümü kullanılarak yapılmış olan şifreleme metotları kullanılmıştır. Dört farklı görselin gerçekleştirilmiş şifreleme ve şifre-çözme işlemleri matematiksel olarak kıyaslanmıştır. Buna göre, çalışmanın sonucunda ayrık dalgacık yöntemi ile birlikte rastgele permütasyon işlemi uygulanarak şifrelenen görüntüler için birden fazla anahtar elde edilerek görüntü güvenliğinin artırıldığı ve şifre çözme işleminin de zorlaştığı gösterilmiştir.

ANAHTAR KELİMELER: Dalgacık Analizi, Görüntü Şifreleme, Rastgele Permütasyon, Görüntü İşleme

TEŞEKKÜR

Bu çalışmanın hazırlanmasında ve kendisinden aldığım derslerin meslek hayatıma katkı sağlamasında üzerimde büyük emeği olan yüksek lisans danışman hocam Dr. Öğr. Üyesi Ahmet Güngör Pakfiliz'e saygılarımı ve teşekkürlerimi sunuyorum.

Annem Zeynep Nilgün Demirci, babam Yusuf Demirci ve abim Ali Tunç Demirci'ye maddi-manevi destekleri için; kız arkadaşım Ülkü Elif Kaya'ya da manevi desteği için teşekkürlerimi sunuyorum.

Dayım Bayram Ergün İşgör'e, bana bu süreçte çalışma imkanı sunarak sağladığı destek için teşekkür ediyorum. Teyzelerim Doç. Dr. Yasemin Gülgün İşgör ve Doç. Dr. Sultan Belgin İşgör'e akademik açıdan yardımları ve manevi destekleri için; anneannem Nebahat İşgör ile dedem Kenan İşgör'e manevi desteklerinden ötürü ayrıca teşekkür ediyorum.

ABSTRACT

Kenan Tuğkan Demirci

GÖRÜNTÜ İLETİMİNDE WAVELET TABANLI RASTELE PERMÜTASYON İLE BİR ŞİFRELEME METODU GELİŞTİRİLMESİ

Baskent University Institute of Science

The Department of Electrical and Electronics Engineering

2020

This study focused on the image information security with better encryption during transfer especially in military defence industry.

In this thesis, 4 different images with different color intensities are used. The image was encrypted once in frequency domain, then in time domain and secured during transfer. In order to ensure the security of image, it will be sufficient to send a sequence vector, which will be difficult to understand the image, along with encrypted image itself to the receiver side is enough to start image decryption process.

While this long vector is transmitted to the receiver side, using LIFO (last in first out) or FIFO (first in first out) algorithms with respect to the pixel quantity of image, will be difficult for the others to decrypt the ciphered image.

In this study, random permutation process in time space and discrete wavelet transform with random permutation process are used as encryption method. The encryption and decryption processes of four different images were compared mathematically. Accordingly, as a result of the study, it has been shown that the image security is increased and the decryption process is difficult by obtaining more than one key for the encrypted images by applying random permutation process with the discrete wavelet method.

KEYWORDS: Wavelet Analysis, Image Encryption, Random Permutation, Image Processing

Bu tezi; araştırma ve geliştirme yaparak, ulu önderimizin bize gösterdiği şekilde ülkemizi daha ileriye taşıyabilmek için çalışan tüm mühendislerine ve çalışanlarına ithaf ediyorum.

K. Tuğkan DEMİRCİ
Ankara - 2020

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

Sayfa

SİMGELER VE KISALTMALAR LİSTESİ

1-D	tek boyutlu (one dimensional)
2-D	iki boyutlu (two dimensional)
A,H,V,D	yaklaşık (approximation), yatay (horizontal), dikey (vertical), köşegen (diagonal) detay
DC	direk akım (direct current)
DWT	ayrık dalgacık dönüşümü (discrete wavelet transform)
F	görüntü matrisi
F(u,v)	iki boyutlu fourier dönüşümü
FIFO	ilk giren ilk çıkar (first in first out)
H	Haar dalgacık dönüştürme matrisi
$H(m)$	entropi değeri
$h_k(z)$	Haar temel fonksiyonları
I(u,v)	F(u,v)'nin imajiner kısmı
LIFO	son giren ilk çıkar (last in first out)
MSE	ortalama kare hatası (mean square error)
$P(m_i)$	m_i 'nin olasılığı
PSNR	tepe sinyal gürültü oranı (peak signal noise ratio)
P(u,v)	F(u,v)'nin güç spektrumu
Υ	korelasyon katsayısı
R(u,v)	F(u,v)'nin reel kısmı
RGB	red (kırmızı), green (yeşil), blue (mavi)
T	Haar dalgacık dönüşüm sonucu matrisi
T(u,v,…)	f(x,y)'nin ayrık dalgacık dönüşümü
XOR	özel "veya" operatörü (exclusive OR)
$\emptyset$(u,v)	F(u,v)'nin faz açısı
$\psi(t)$	Haar dalgacığı
$\psi(x,y)$	iki boyutlu dalgacık fonksiyonu
$\psi^H(x,y)$	yatay dalgacık
$\psi^V(x,y)$	dikey dalgacık
$\psi^D(x,y)$	köşegen dalgacık
$\varphi(x,y)$	iki boyutlu ölçeklendirme fonksiyonu

1. GİRİŞ

Günümüzde görüntüler üzerinde şifreleme işlemi savunma, tıp, güvenlik gibi birçok alanda kullanılmaktadır. Bu çalışmalarda kullanılan algoritmalar arasında doğruluk, zaman, hata-kayıp gibi birçok konu ele alınmaktadır.

İletişim gizliliğinin sağlanabilmesi için şifreleme yöntemleri kullanılmaktadır. Özellikle askeri alanda verilerin düşmana veya ulaşmaması gereken yerlere karşı güvenliğinin sağlanması, çeşitli şifreleme yöntemleriyle icra edilmektedir. Ele geçmesinin istenmediği veriler; bir şifre, koordinat, ses, video veya görüntü olabilmektedir.

Genellikle insansız hava araçlarında, canlı ve arşiv görüntülerine olan erişimin güvenliğini sağlamak için otomatik olarak geçici şifreler kullanılarak görüntü sunucusuna olan erişimin güvenliği sağlanmaktadır [1].

Literatürde benzer çalışmalarda uygulanan şifreleme yöntemleri, veri sıkıştırma, veri güvenliğinin sağlık gibi alanlara hizmet etmek amacıyla gerçekleştirildiği görülmektedir [2]. Bahsi geçen bu çalışmalarda daha kapsamlı ve özellikle savunma sanayinde kullanılabilecek güvenlikte çeşitli yöntemlerde çalışmalar mevcuttur. Literatürde genellikle görsellerin satır ve sütun pikselleri karıştırılarak [3] veya görsellerin karelere bölünmesiyle [4, 5] anlamsızlaştırılması üzerine çalışmalar mevcuttur. Bu çalışmalar, hızları bakımından avantaj sağlamakla birlikte şifreleme-şifre çözme işlemleri sadece tek bir anahtar ile çözülebildiğinden bu bakımdan da mevcut çalışmanın literatürde yer alan çalışmalara kıyasla farklılığı söz konusudur.

Şifreleme işlemi için aynı görüntü üzerinde; hem rastgele permütasyon kullanılarak zaman uzayında şifrelenmesi, hem de ayrık dalgacık dönüşümü uygulanarak frekans uzayında şifrelenmesi ele alınarak incelenecektir. Ayrık dalgacık dönüşümü uygulanarak şifrelenmiş bir görüntünün, tekrar anlaşılır haline geri döndürülmesi hedeflenmiştir. Bu şekilde üçüncü şahıslara karşı verinin gizliliği sağlanmış olacaktır. Bu uygulamanın, özellikle savunma alanında kullanılabilmesi öngörülmektedir.

1.1. Konunun Öncesi

Askeri alanda gönderilen şifreli verinin karşı taraftan daha zor bir şekilde anlaşılması sağlayacak algoritmaların nasıl yapılabileceği üzerinde çalışmalara başlanmıştır. Bu çalışmalarda video ve görüntü gibi verilerin şifreleme metotlarının geliştirilerek daha güvenli bir veri aktarımı gerçekleştirilmesi hedeflenmiştir. Görüntünün şifreleme işleminden sonra veya yüksek çözünürlüğe sahip olduğu için hafızada kapladığı boyutunun büyük olmasından dolayı zaman uzayından farklı bir uzayda işlem yapılması gerektiği düşünülmüştür. Bu şekilde görüntü sıkıştırma işlemi gerçekleştirilmiş ve boyutun indirgenmiş olacağı, fakat görüntü kalitesinden de ödün verileceği görülmüştür.

Görüntü şifreleme ile ilgili yapılan çalışmalar arasında piksel konumlarının değiştirilmesi ile görüntülerin şifrelenmesi sağlanmaktadır [2, 4]. Bu şifreleme işlemi, hiçbir dönüşüm yapılmaksızın direkt olarak zaman uzayında gerçekleştirildiğinden şifrenin tekrar çözülmesini de kolaylaştıracaktır.

2016 yılında yapılmış bir çalışmada; RGB görüntülerin şifrelenmesi için rastgele bir çözüm anahtarı oluşturulması ve bu işlemin en kısa sürede icra edilmesi için XOR işlemi kullanılarak şifre çözümünün tahmin edilebilirliğini güçleştirmiştir. Dalgacık dönüşümü ile R, G ve B kanallarındaki detaylar elde edilmiştir. Görüntü farklı konumlardan satır ve sütunlara bölünerek elde edilen parçaların orijinleri etrafında döndürülmelerinden sonra XOR mantığı ile rastgele permütasyon işlemi uygulanmıştır. Tekrar bu görüntü parçaları orijinleri etrafında döndürülüp XOR işlemi tekrar uygulanmış ve tüm bu işlemler R, G ve B kanalları için ayrı ayrı gerçekleştirilmiştir. Ters dalgacık dönüşümü ile birlikte şifreli görüntü çözüm anahtarları kullanılarak orijinal anlamlı görüntü elde edilmiştir [2].

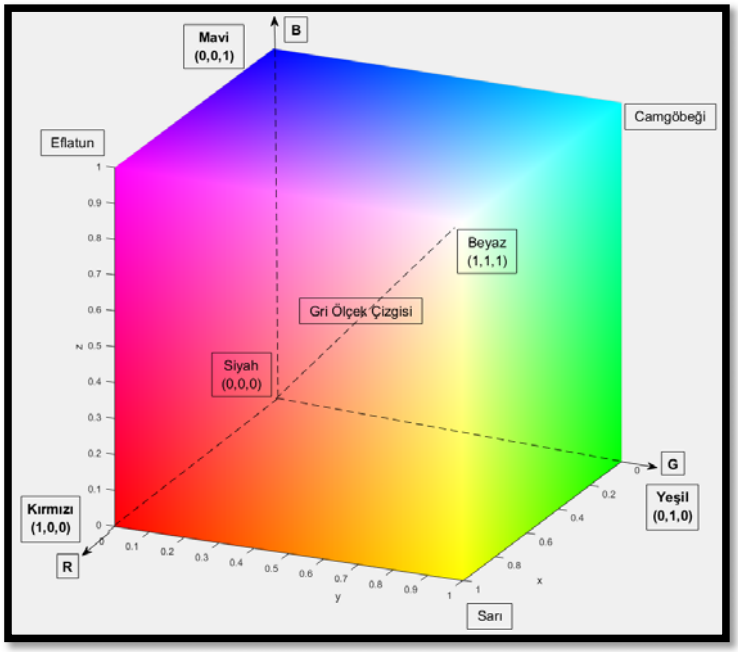
Yapılan diğer çalışmalar arasında XOR (Exclusive OR) kullanılarak şifrelenmiş görüntünün tekrar eski kalitesine yakınsaması [6], kaotik lojistik haritalar ile birlikte ayrık dalgacık dönüşümü kullanılarak görüntülerin şifrelenmesi [7], polinom yakınsaması ve matematiksel işlemler kullanılarak şifrelendirilmiş renkli görüntülerin rastgele dizi ile paylaşım güvenliğinin sağlanması [8], kaotik lojistik harita ile birlikte görüntünün bloklara ayrıştırılması sağlanarak homojen konum değişikliği yapılması [5], sağlık alanında elde edilen görsellerin aktarımı esnasında güvenliğin sağlanması için piksel konumlarının yer değiştirilerek görselin şifrelenmesi [3], görüntüleri şifreleme teknikleri üzerine yapılmış çeşitli çalışmalar [9-11] ve bu çalışmalara benzer diğer çalışmaların da [12] yapıldığı

gözlemlenmiştir. Çoğu çalışmada dalgacık analizi ile görüntü sıkıştırması gerçekleştirilmiş, çoğunlukla zaman uzayında şifreleme işlemi yapıldığı görülmüştür. Bu çalışmaların yanısıra, farklı metotlar uygulanarak renkli görüntülerin şifrelenmesi üzerine yazılmış kaynaklar da incelenmiştir [13, 14].

1.2. Teorik Altyapı

Herhangi bir görseldeki keskinlik, derinlik, doygunluk gibi birçok detay, objelerin renkleri ile daha kolay anlaşılabilmektedir.

Görüntüler, matematiksel olarak bir matris olarak ifade edilmektedirler. Çekilen bir fotoğrafın çözünürlüğü, görüntünün M x N matrisini oluşturmaktadır. Görüntü eğer renkli, yani RGB (red-green-blue) bir yapıdaysa bu matris M x N x 3 olarak tanımlanmaktadır. Görselin her bir pikseli matrisin elemanlarıyla ilişkilidir.



Şekil 1.1. RGB Birim Kartezyen Renk Modeli

RGB modelinde (Şekil 1.1), her bir renk kırmızı, yeşil ve mavinin birincil spektral bileşenlerinde görünür. Kırmızı, yeşil ve mavi bileşenlerin sıfır olduğu orijin noktası siyah; üç bileşenin de 1 olduğu nokta ise beyaz renk olarak RGB birim kartezyen modelinde ifade edilir [15].

Orijin noktası ile RGB değerlerinin 1 olduğu beyaz noktayı birleştiren çizgi, gri ölçeği temsil eder. Bu çizgi üzerindeki kırmızı, yeşil ve mavi değerler birbirine eşittir.

Renkli bir görüntüde yapılacak işlem yoğunluğu, tek boyutlu, yani gri ölçekli bir görsele nazaran çok daha fazladır. Bu nedenle ilgili görüntü üzerinde yapılacak işlemler gri ölçekli görsel üzerinden yapılmaktadır.

1.3. Dönüşüm

Görüntü işleme; sayısal görüntülerin çeşitli dönüşümler kullanılarak farklı uzaylarda uygulanan gürültü yok etme, kontrast geliştirme, keskinleştirme, segmentasyon ve algılama gibi matematiksel işlemlerin bütünüdür. Özellikle resimlerle ilişkili bir sinyal işleme metodudur.

Görüntü iyileştirme işlemleri, iki farklı uzayda gerçekleştirilebilmektedir. Zaman uzayında (spatial domain) görüntü pikselleri üzerinde direk işlemler uygulanarak kontrast, keskinleştirme, gürültü yok etme gibi basit matematik işlemleri ile sonuca ulaşılır.

Frekans uzayında gerçekleştirilen görüntü işleme metotlarında butterworth, gaussian gibi alçak geçiren, yüksek geçiren, bant durduran veya bant geçiren filtreleme işlemleri yapılmaktadır.

Zaman uzayından frekans uzayına Fourier dönüşümü kullanılarak geçilir. Fourier dönüşümünde bir sinyal, zaman dizisi yerine frekans dizisi ile ifade edilir. Bu sinyal, farklı frekansların oluşturduğu bir sinyal olarak görülür. Zaman uzayındaki bir sinyal, zaman-genlik (şiddet) grafiği kullanılarak ifade edilirken, Fourier dönüşümünden sonra bu sinyal frekans ortamında frekans-genlik grafiği kullanılarak gösterilir.

Fourier dönüşümü sonrasında zaman uzayından frekans uzayına geçiş sağlanır ve zaman bilgisi kaybolur. Sinyal üzerindeki değişikliğin ne zaman olduğu bilgisi de elde edilmez. Ters Fourier dönüşümü uygulanarak sinyal tekrar zaman uzayına geçirilir ve zaman bilgisi elde edilir. Fourier dönüşümü sinyali, sinüs ve kosinüs fonksiyonlarının doğrusal bileşimi şeklinde ayrıştırır.

Sinyal üzerinde yapılacak filtreleme gibi işlemlerin zaman uzayında gerçekleştirilmesi çok zor, hatta karmaşık sinyallerde imkansıza yakındır. Bu nedenle, sinyal üzerinde Fourier dönüşümü ile zaman uzayından frekans uzayına geçiş sağlanıp, gerekli işlemler yapıldıktan sonra tekrar zaman uzayına geçiş sağlanmaktadır.

Frekans uzayında görüntünün frekans bileşenleri birbirinden ayırt edilebilmektedir. Böylece yüksek geçiren, bant geçiren ve alçak geçiren gibi filtrelerin uygulaması kolaylıkla yapılabilmektedir.

Görüntüler Fourier dönüşümünden sonra frekans uzayına geçirilmiş olur. Dönüşüm sonucunda görüntünün yüksek frekansları kenarlarda, alçak frekansları ise orta kısımlarda toplanır. Zaman uzayında yatay olan çizgiler dikey, dikey olan çizgiler ise yatay olarak görüntülenir.

1.3.1. Fourier dönüşümü

Fourier dönüşümü; görüntü geliştirme, iyileştirme, veri sıkıştırma gibi işlemlerin yapılabilmesi için zaman uzayından frekans uzayına geçişi sağlayan bir dönüşümdür.

$f(x,y)$ için $x = 0, 1, 2, \dots, M-1$ ve $y = 0, 1, 2, \dots, N-1$ olmak üzere iki boyutlu M x N kadar pikseli olan dijital bir görüntüyü ifade etmektedir. $f(x,y)$'nin iki boyutlu ayrık Fourier dönüşümü

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{1.1}$$

$u = 0, 1, 2, \dots, M-1$ ve $v = 0, 1, 2, \dots, N-1$ şeklinde ifade edilebilir.

Frekans uzayı, $F(u,v)$ tarafından u ve v frekans değişkenlerini kapsayan bir koordinat sistemidir.

Ters ayrık Fourier dönüşümü ise

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{1.2}$$

şeklinde ifade edilebilir. Bu dönüşüme Fourier katsayıları açılımı da denilmektedir.

Dönüşümdeki frekans uzayının orijin noktası olan $F(0,0)$, Fourier dönüşümün doğru akım bileşeni (dc component) olarak ifade edilir ve sıfır frekans değerindeki akımı temsil eder.

Eğer $f(x,y)$ dizisi reel ise Fourier dönüşümü sonucu kompleks sayılar elde edilir. Fourier dönüşümü uygulanmış bir görselin; frekans uzayındaki görsel analizi, görselin spektrumunu ifade eder.

$F(u,v)$'nin genliği;

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)} \tag{1.3}$$

$F(u,v)$'nin faz açısı;

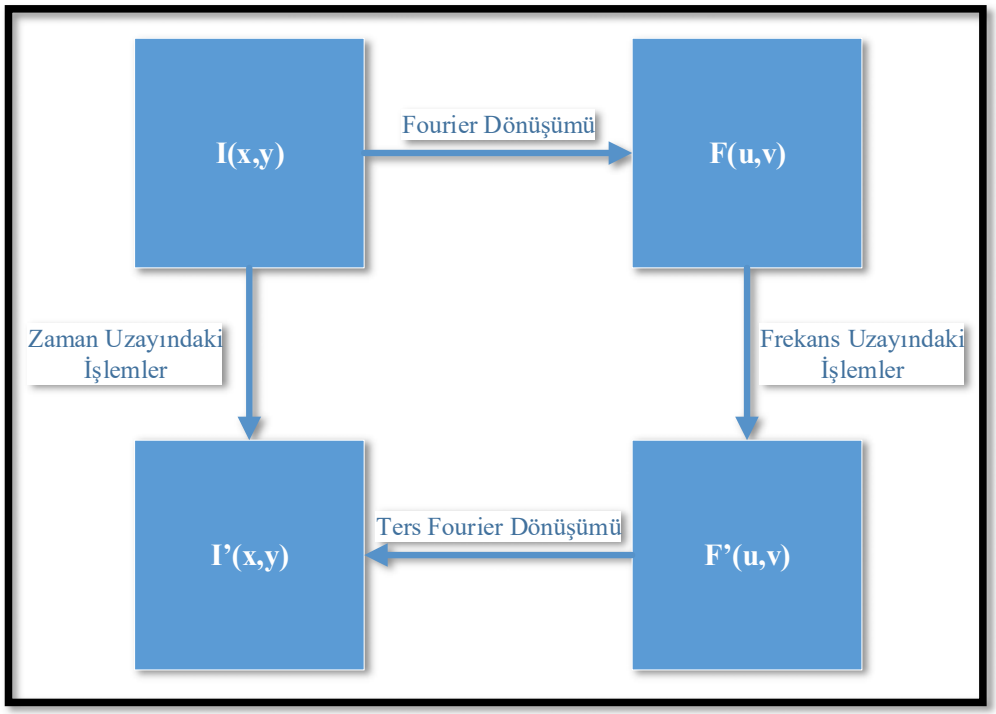
$$\emptyset(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \tag{1.4}$$

güç spektrumu;

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v) \tag{1.5}$$

şeklinde ifade edilir.

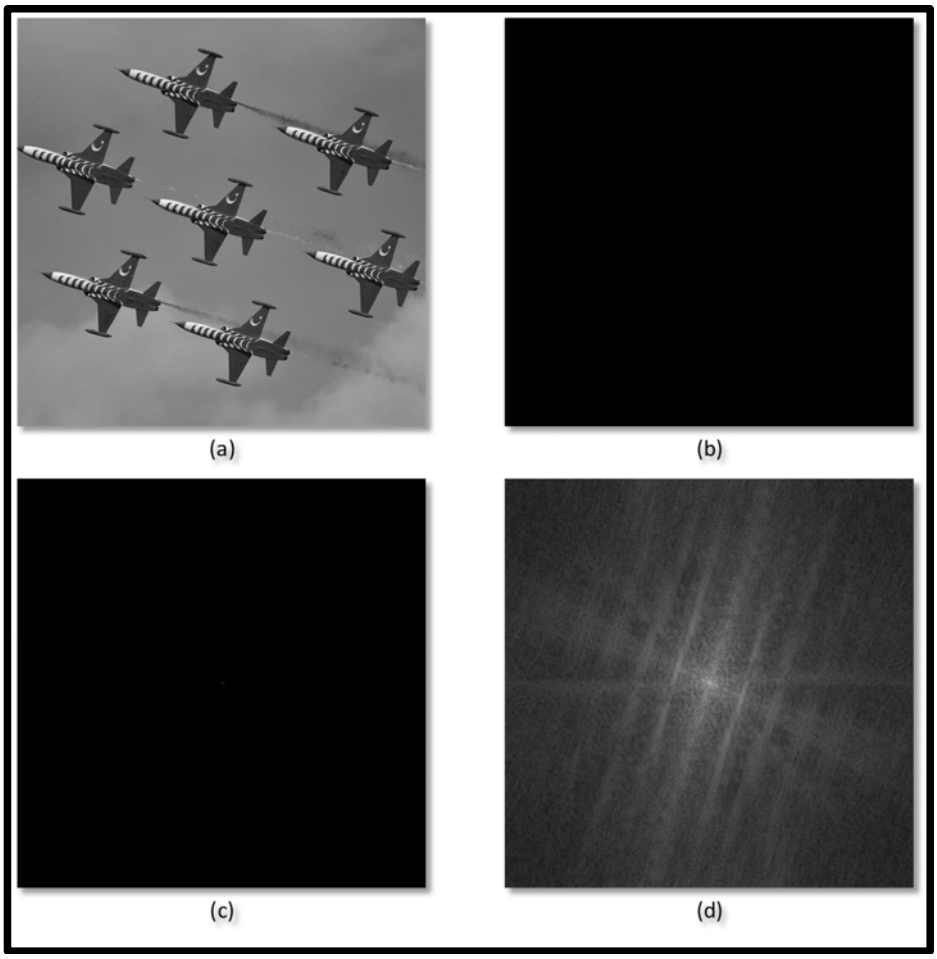
$R(u,v)$ ve $I(u,v)$, $F(u,v)$'nin sırasıyla reel ve imajiner kısımlarını gösterir. Fourier veya güç spektrumu ile bir görsel, frekans uzayında ifade edilebilir. Görüntünün Fourier dönüşümü sonrasında frekans uzayına geçişi sağlanır. Frekans uzayındaki görüntünün yüksek frekansları kenarlarda, alçak frekansları ise orta kısımlarda toplanır. Frekans uzayındaki bu görselin tekrar zaman uzayına dönüştürülebilmesi için hem genlik hem de faz bilgisine ihtiyaç duyulur.



Şekil 1.2 Frekans Uzayında Görüntü İşleme Aşamaları

$f(x,y)$ görüntüsünden elde edilen $F(u,v)$ Fourier katsayılar matrisinin elemanları kompleks sayılardır ve bu nedenle direk olarak zaman uzayında görüntülenebilmesi mümkün değildir. $F(u,v)$ matrisinin genlikleri alınarak $|F(u,v)|$ double (noktalı format) tipinde sayılar elde edilir. $|F(u,v)|$ matrisindeki en büyük değer, yani DC değeri (m) bulunur. $|F(u,v)|/m$ işlemi gerçekleştirildikten sonra görsel görüntülenebilir.

Fourier spektrumundaki aralık $[0,10^6]$ gibi çok büyük bir aralığa sahiptir. Bu nedenle 8-bit olarak görüntülemek istenildiğinde büyük değerler baskın olacaktır. Bunun sonucu olarak görüntünün düşük spektrum değerlerinde kayıplar meydana gelecektir. $g = c * \log(1+f)$ formülü hesaplanarak dinamik uzaklık (dynamic range) indirgenebilir (c: sabit, f: küsurat, g: logaritmik dönüşüm). DC değerinin çok büyük olmasından dolayı bu değer baskın çıkacaktır. Bu nedenle $\log(1+|F(u,v)|)$ logaritmik dönüşümü uygulanarak dinamik uzaklık indirgenir (Şekil 1.3).



Şekil 1.3. Gri Ölçekli Görsel (a), Fourier Spektrum (b), Merkezlenmiş Spektrum (c), Logaritmik Dönüşüm ile Görselleştirilmiş Spektrum (d)

Fourier dönüşümünün mutlak değerinin (genliğinin) görüntülenmesi ile Fourier dönüşümün spekturumu alınmış olur [16, 17].

1.3.2. Ayrık dalgacık (wavelet) dönüşümü

Dalgacıklar, sinyallerin veya görüntülerin birden fazla çözünürlükte analiz edilmesini ve gösterilmesini sağlarlar.

Dijital görüntüler, birden çok çözünürlükte görüntüleneceği veya işleneceği zaman ayrık dalgacık analizi kullanılabilir. Ayrık dalgacık analizi kullanılarak gerçekleştirilen işlemler; verimliliklerinin yanısıra, görüntülere ait mekânsal ve frekans özellikleri hakkında güçlü veriler elde etmeyi sağlar. Dalgacık dönüşümü, Fourier dönüşümünde elde edilen

frekans bilgisine ilaveten zaman bilgisini de sağlamaktadır. Bu şekilde sinyalin, hangi zamanda hangi frekans bilgisini içerdiği incelenebilmektedir.

Ayrık dalgacık dönüşümü; sinyali filtreler aracılığıyla, farklı frekans bantları içinde inceleyen ve sonuçları yaklaşık (approximation), yatay (horizontal), dikey (vertical) ve köşegen (diagonal) detay katsayı dizilerine ayıran bir dönüşümdür. Bu dönüşüm sonrasında dijital verilerden elde edilen katsayılar matrisler ile ifade edilir. Bu matris verileri, orijinal görselin matris verilerine göre farklılık gösterir. Sıkıştırma, gürültü temizleme ve şifreleme gibi işlemlerde bu dalgacık dönüşümü kullanılarak hem zaman hem frekans bilgileri aynı anda incelenebilmektedir [18].

İki boyutlu M x N boyutunda bir görüntüyü ifade eden $f(x,y)$ için ayrık dalgacık dönüşüm aşağıdaki şekilde ifade edilebilir:

$$T(u,v,\dots) = \sum_{x,y} f(x,y)\, g_{u,v,\dots}(x,y) \tag{1.6}$$

Buradaki x ve y mekansal değişkenleri; u,v,.. ise dönüşüm uzayı değişkenleridir. Aşağıdaki eşitlikte de gösterildiği üzere, bahsi geçen görüntüye ait ayrık dönüşümü ifade eden $T(u,v,\dots)$ matrise, ters ayrık dönüşüm uygulanarak M x N boyutundaki görüntü $f(x,y)$ tekrar elde edilebilmektedir.

$$f(x,y) = \sum_{u,v,\dots} T(u,v,\dots)\, h_{u,v,\dots}(x,y) \tag{1.7}$$

Eşitlik (1.6)'da verilen $g_{u,v,\dots}(x,y)$ ve eşitlik (1.7)'deki $h_{u,v,\dots}(x,y)$ ise sırasıyla ileri ve ters dönüşüm çekirdeklerini (kernel) ifade eder. Eşitlik (1.6)'daki $T(u,v,\dots)$ olarak belirtilen dönüşüm katsayıları, f'in $\{h_{u,v,\dots}\}$'ye göre seri açılım katsayıları olarak da görülebilir. Ayrık Fourier dönüşümü, seri açılım formülüne uymaktadır. Buna göre;

$$h_{u,v}(x,y) = {g^*}_{u,v}(x,y) = \frac{1}{\sqrt{MN}}\, e^{j2\pi(ux/M+vy/N)} \tag{1.8}$$

$j = \sqrt{-1},\ u = 0,1,\dots\ ,M-1, v = 0,1,\dots\ ,N-1$ ve * işareti ise karmaşık eşleniği ifade etmektedir. Dönüşüm uzayı değişkenleri olan u ve v, sırasıyla yatay ve dikey frekanslarını ifade etmektedir. Bu çekirdekler şu şekilde ayrıştırılabilir;

$$h_{u,v}(x,y) = h_u(x)h_v(y) \tag{1.9}$$

$$h_u(x) = \frac{1}{\sqrt{M}} e^{j2\pi ux/M} \tag{1.10}$$

$$v(y) = \frac{1}{\sqrt{N}} e^{j2\pi vy/N} \tag{1.11}$$

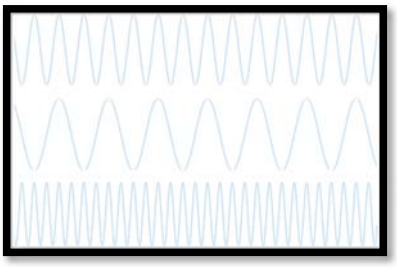
ve aşağıdaki formüle göre de ortonormallik göstermektedir:

$$< h_r,\, h_s > = \delta_{rs} = \begin{cases} 1, & r = s \\ 0, & diğer\ durumlarda \end{cases} \tag{1.12}$$

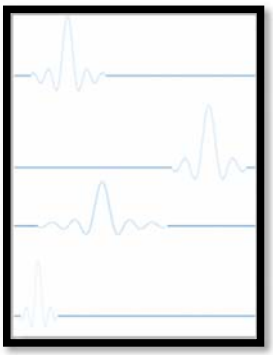
< > operatörleri iç çarpımı nitelemektedir. İki matrisin veya vektörün iç çarpımı için, her bir eleman birbiriyle çarpılarak toplanır. Geometrik olarak ortogonalliğin (dikliğin) belirlenebilmesi için bu çarpım sonucunun sıfıra eşit olması gerekmektedir. Ortonormallik için bu vektörlerin her birisi birbirine dik ve vektör uzunlukları 1 olmalıdır. Çekirdeklerin bu şekilde ayrıştırılabilir olması 2-D dönüşümünü, 1-D dönüşümünde kullanılan satır-sütun ya da sütun-satır geçişlerini sağlayarak kolaylaştırır. Ortonormallik sayesinde, ileri ve ters dönüşüm çekirdekleri birbirinin karmaşık eşleniği olur (ileri ve ters dönüşüm çekirdekleri fonksiyonlar reel olduğunda aynıdır).

Dalgacık dönüşümleri; temel fonksiyonları sinüzoid olan Fourier dönüşümünün aksine, dalgacık adı verilen ve değişen frekansta, sınırlı süreli küçük dalgalara dayanır. Fourier dönüşümü görüntülere ait sadece frekans özellikleri hakkında bilgi verir [19, 20].

Ayrık dalgacık dönüşümü, ayrık Fourier dönüşümünden farklı olarak tek bir dönüşüm çekirdek çifti etrafında dönen iki basit denklem ile ifade edilir. Ayrık dalgacık dönüşümü, kullanılan dönüşüm çekirdeklerinde farklılık göstermeyen, aynı zamanda bu işlevlerin temel doğasında da farklılık gösteren bir sınıf dönüşümleri ifade eder. Ayrık dalgacık dönüşümü benzersiz olmakla birlikte dönüşüm çeşitliliği içerir ve tek bir denklem ile dönüşümler tamamen açıklanamaz. Bunun yerine her bir ayrık dalgacık dönüşüm, dönüşüm çekirdek çiftleriyle veya bu çiftleri tanımlayan parametreler seti ile karakterize edilir. Çeşitli dönüşümlerin dalgacık olarak nitelendirilmeleri; genişleme fonksiyonlarının, değişen frenkans (Şekil 1.3) ve sınırlı süreli küçük dalgalar (Şekil 1.4) olmasına dayanmaktadır [15].



Şekil 1.4. Fourier Açılımı (Değişen Frekans ve Sonsuz Süreli Sinuzoidler)



Şekil 1.5. Ayrık Dalgacık Açılımı (Sonlu Süre ve Değişen Frekanstaki Küçük Dalgalar)

Bu dalgacıkların hepsinde ortak olarak görülen özellikler aşağıdaki gibidir.

Özellik 1: Ayrılabilir, Ölçeklenebilir, Çevrilebilir

Çekirdekler, üç adet 2-D dalgacıklar halinde gösterilebilir.

$$\psi^H(x,y) = \psi(x)\varphi(y) \tag{1.13}$$

$$\psi^V(x,y) = \varphi(x)\psi(y) \tag{1.14}$$

$$\psi^D(x,y) = \psi(x)\psi(y) \tag{1.15}$$

$\psi^H(x,y)$: Yatay dalgacık

$\psi^V(x,y)$: Dikey dalgacık

$\psi^D(x,y)$: Köşegen dalgacık

2-D ölçeklendirme fonksiyonu ise aşağıdaki şekilde gösterilir.

$$\varphi(x,y) = \varphi(x)\varphi(y) \tag{1.16}$$

Her bir 2-D fonksiyon, iki adet 1-D ölçekleme ve dalgacık fonksiyonunun bir çarpımıdır.

$$\varphi_{j,k}(x) = 2^{j/2}\varphi\left(2^j x - k\right) \tag{1.17}$$

$$\psi_{j,k}(x) = 2^{j/2}\psi\left(2^j x - k\right) \tag{1.18}$$

k : 1-D fonksiyonların x ekseni üzerindeki pozisyonu

j : 1-D fonksiyonların x ekseni boyunca ne kadar dar veya geniş olduğu

$2^{j/2}$: yükseklik veya genlik bilgisinin kontrolü

Ana dalgacık (1.19) ve ölçekleme (1.20) fonksiyonları ise aşağıdaki şekilde ifade edilir.

$$\psi(x) = \psi_{0,0} \tag{1.19}$$

$$\varphi(x) = \varphi_{0,0} \tag{1.20}$$

Özellik 2: Çoklu Çözünürlük Uyumluluğu

1-D ölçeklendirme fonksiyonu, çoklu çözünürlük analizi için aşağıdaki gereksinimleri karşılar:

a. $\varphi_{j,k}$ tam sayı değerlerine karşı ortogonallik (diklik) gösterir.
b. Küçük ölçeklerde veya çözünürlüklerde $\varphi_{j,k}$ serisinin genişlemesi olarak temsil edilebilecek fonksiyonlar kümesi (örneğin j değeri), büyük ölçeklerde temsil edilebilecek fonksiyonlar içinde bulunur.

Yukarıda belirtilen koşullar sağlandığında; tamsayı çevirileri ve ikili ölçeklendirmeleri ile birlikte, bitişik ölçeklerde $\varphi_{j,k}$ tarafından temsil edilebilir herhangi iki işlev kümesi arasındaki farkı kaplayan tamamlayıcı bir dalgacık $\psi_{j,k}$ vardır.

Özellik 3: Ortogonallik

Genişletme fonksiyonları, 1-D ölçülebilir, kare ile birleştirilebilir fonksiyonlar kümesi için ortonormal veya biortogonal (çokgen) şeklinde bir temel oluşturur. Temel olarak

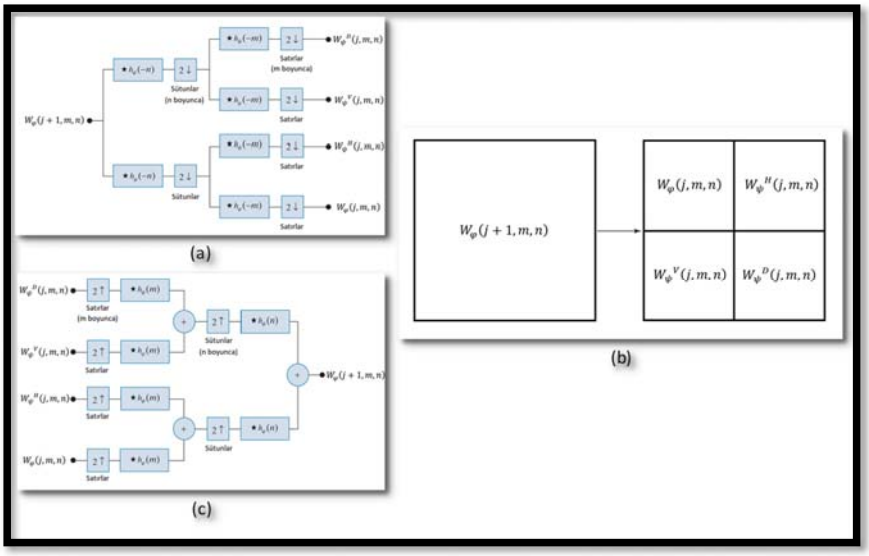
adlandırılabilmesi için, temsil edilebilir her fonksiyon için benzersiz bir genişleme katsayıları kümesi olmalıdır. Reel ortonormal çekirdekler için $g_{u,v\ldots} = h_{u,v\ldots}$ eşitliği geçerli iken, biortogonal durumda aşağıdaki formül geçerlidir.

$$< h_r,\ g_s > = \delta_{rs} = \begin{cases} 1, & r = s \\ 0, & diğer\ durumlarda \end{cases} \tag{1.21}$$

g : h'nin çifti

Ölçeklendirme ve dalgacık fonksiyonlarına sahip biortogonal bir dalgacık dönüşümü için $\varphi_{j,k}(x)$ ve $\psi_{j,k}(x)$'in çiftleri; $\tilde{\varphi}_{j,k}(x)$ ve $\tilde{\psi}_{j,k}(x)$ şeklinde ifade edilir.

İki boyutlu dalgacık dönüşümünün gerçekleştirilmesi için iki boyutlu ölçeklendirme fonksiyonu $\varphi(x,y)$ ve üç adet iki boyutlu dalgacık $\psi^H(x,y)$, $\psi^V(x,y)$, $\psi^D(x,y)$ gerekmektedir. Bu dalgacıklar kullanılarak görünülerin farklı yönlerindeki yoğunluk değişimleri ölçülür. $\psi^H(x,y)$ görüntünün sütunlarındaki değişimi ölçerek yatay kenarları ortaya çıkarır. $\psi^V(x,y)$ görüntünün satırlarındaki değişimi ölçerek dikey kenarları ortaya çıkarır. $\psi^D(x,y)$ ise görüntünün köşegenlerindeki değişimi ölçer. İki boyutlu dalgacık dönüşümü dijital filtreler ve alt örneklemleyiciler (downsampler) ile ifade edilebilir (Şekil 1.6).



Şekil 1.6. 2-D Dalgacık Dönüşümü (a), Dönüşüm Sonucu Ayrışma (b), Ters Dalgacık Dönüşümü (c)

$f(x,y)$ görüntüsüne yüksek ve alçak geçirgen filtreler uygulanır. Yüksek geçirgen filtre sonucunda görüntünün detay katsayıları elde edilirken; alçak geçirgen filtre çıktısında ise yaklaşık (approximation) katsayıları elde edilir. Filtreler uygulandıktan sonre elde edilen kaysayılar alt örnekleme (downsampling) yapılarak yarıya düşürülür. Bu işlem hem sütunlar hem de satırlar için uygulanır.

Yukarıda belirtilen özelliklerin sonucu hem $\varphi(x)$ hem de $\psi(x)$ kendilerinin iki boyuttaki doğrusal kombinasyonları olarak seri açılımları ile ifade edilebilmeleridir.

$$\varphi(x) = \sum_{n} h_{\varphi}(n)\sqrt{2}\,\varphi(2x-n) \tag{1.22}$$

$$\psi(x) = \sum_{n} h_{\psi}(n)\sqrt{2}\,\varphi(2x-n) \tag{1.23}$$

h_{φ} ve h_{ψ} açılım katsayıları aynı zamanda sırasıyla ölçekleme ve dalgacık vektörleri olarak tanımlanmaktadır. Bu katsayılar dalgacık dönüşümü filtre katsayılarıdır. $W_{\varphi}(j,m,n)$ ve ${W_{\psi}}^{H}(j,m,n)$, ${W_{\psi}}^{V}(j,m,n)$, ${W_{\psi}}^{D}(j,m,n)$ çıktıları, j ölçeklemesindeki ayrık dalgacık dönüşümü katsayılarını ifade eder.

Şekil 1.6'daki $h_{\varphi}(-n)$ ölçekleme ile $h_{\psi}(-m)$ dalgacık vektörleri sırasıyla alçak geçirgen ve yüksek geçirgen ayrıştırma filtreleridir. Şekil 1.6'da gösterilen aşağı ok işareti ile birlikte yazılmış 2 rakamı ise, bir dizi noktadan diğer her bir noktayı çıkartmak için alt örnekleme (downsampling) işlemini ifade etmektedir.

$${W_{\psi}}^{H}(j,m,n) = h_{\psi}(-m) * \left[h_{\varphi}(-n) * W_{\varphi}(j+1,m,n)\big|_{n=2k,k\geq 0}\right]\Big|_{m=2k,k\geq 0} \tag{1.24}$$

Herhangi bir dalgacık katsayısına ulaşılması için gerekli işlemlerde geçen yıldız (*) işareti konvolüsyonu temsil eder. Filtreleme ve 2 ile alt örnekleme işlemi, negatif olmayan çift indislerde konvolüsyon yapılır ve dalgacık katsayısına ulaşılır.

Şekil 1.6'daki $W_{\varphi}(j+1,m,n)$ girişinden 4 farklı düşük ölçekteki katsayı matrisleri elde edilir. W_{φ}'nin katsayılarını bulmak için iki kez alçak geçirgen filtre uygulanır ve bu katsayıya da yaklaşık (approximation) katsayısı elde edilir. ${W_{\psi}}^{H}(j,m,n)$, ${W_{\psi}}^{V}(j,m,n)$ ve ${W_{\psi}}^{D}(j,m,n)$ çıktıları ise sırasıyla yatay (horizontal), dikey (vertical) ve köşegen (diagonal) detay katsayılarıdır.

Orijinal görüntü $f(x, y)$, Şekil 1.6'daki ilk iterasyon için giriş ile aynıdır. Birden fazla iterasyon yapılırken $W_{\varphi}(j, m, n)$ yaklaşık katsayısı, ikinci iterasyonun girişi olarak tanımlanır ve her iterasyon sonucunda ortaya çıkan yaklaşık katsayısı, yapılacak olan yeni iterasyonun girişi olmaya devam eder.

1.3.2.1. Haar dalgacık dönüşümü ve diğer dalgacıklar

Dalgacık dönüşümleri yapılırken temel bir dalgacık operatörü ile dönüşüm gerçekleştirilir. Bu dalgacık operatörleri birbirleriyle aynı özelliklere sahip değildir. Değişen derecelerine bağlı olarak sahip oldukları birleşme noktası sayısı, genişliği, ortogonalliği, ölçek fonksiyonu bulundurup bulundurmama, kompleks olma gibi farklar barındırmaktadırlar.

Ayrık veya sürekli dalgacık dönüşümlerinde kullanılan dalgacık operatörlerinin bazıları, hem ayrık hem de sürekli dalgacık dönüşümlerinde kullanılabilmektedir.

Sürekli dalgacıklar için Haar, Daubechies, Symlets, Coiflets, Biorthogonal, Gauss, Mexican Hat, Morlet, Meyer, Shannon dalgacıkları birer örnek olarak verilebilir. Ayrıca Haar, Daubechies, Symlets, Coiflets, Biorthogonal dalgacıkları, ayrık dalgacık dönüşümlerine de uygun dalgacık operatörleri olarak gösterilebilir.

1909 yılında bulunmuş ilk dalgacık olan Haar dalgacığı, Daubechies dalgacığının en az birleşme noktasına sahip olduğu, 1. derecedeki Daubechies dalgacık tipidir. Sadece 1 adet birleşme noktası bulundurduğundan düzgün ve yumuşak geçişli fonksiyonlar için uygun bir sinyal değildir [20].

Haar dalgacığı kendi başına ayrıştırılabilen, simetrik olan, en eski ve basit dalgacık türüdür. Matris formunda aşağıdaki şekilde ifade edilebilir.

$$T = HFH \tag{1.25}$$

F : NxN görüntü matrisi

H : NxN dönüştürme matrisi

T : NxN dönüşüm sonucu matrisi

Haar dönüşümü için H dönüşüm matrisi, $h_k(z)$ olarak nitelendirilen Haar temel fonksiyonlarını içerir. Haar temel fonksiyonları;

$z \in [0,1]$, $k = 0, 1, 2, \dots, N-1$ ve $N = 2^n$ şeklinde sürekli, kapalı bir aralıkta tanımlanırlar.

H matrisini üretmek için k tamsayısı aşağıdaki şekilde tanımlanır.

$$k = 2^p + q - 1 \tag{1.26}$$

$$0 \le p \le n-1,\ q = 0\ ya\ da\ q = 1,\ 1 \le q \le 2^p\ ,\ p \ne 0$$

Buna göre Haar temel fonksiyonları aşağıdaki şekilde tanımlanır.

$h_0(z) = h_{00}(z) = \frac{1}{\sqrt{N}}\ , z \in [0,1]$

$$h_k(z) = h_{pq}(z) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2}, & (q-1)/2^p \le z < (q-0.5)/2^p \\ -2^{p/2}, & (q-0.5)/2^p \le z < q/2^p \\ 0, & diğer\ durumlarda, z \in [0,1] \end{cases} \tag{1.27}$$

NxN Haar dönüşüm matrisinin i'nci satırı elemanlarını $z = \frac{0}{N}, \frac{1}{N}, \frac{2}{N}, \dots, (N-1)$ olacak şekilde $h_i(z)$ içermektedir.

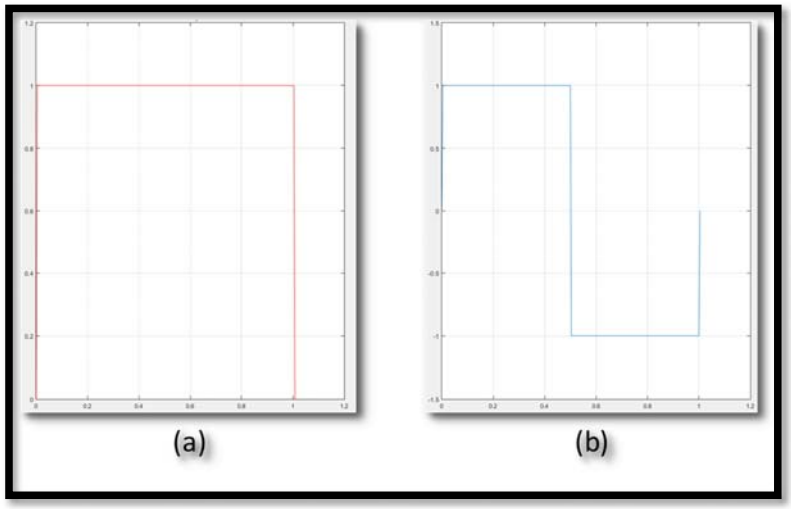
Haar dalgacığı aşağıdaki şekilde ifade edilir.

$$\psi(t) = \begin{cases} -1, & 0 \le t < \frac{1}{2} \\ 1, & \frac{1}{2} \le t < 1 \\ 0, & diğer\ durumlarda \end{cases} \tag{1.28}$$

Ölçeklendirme fonksiyonu $\varphi = 1_{[0,1]}$ ve $h[n]$ filtresi $2^{-1/2}$ değerine eşit, $n = 0$ ve $n = 1$ olmak üzere sıfırdan farklı iki adet katsayıya sahipken aşağıdaki şekilde Haar dalgacığı elde edilebilir [17, 20, 21].

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{\infty} (-1)^{1-n} h[1-n]\varphi(t-n) = \frac{1}{\sqrt{2}}\big(\varphi(t-1) - \varphi(t)\big) \tag{1.29}$$

Haar dalgacığının ölçeklendirme ve dalgacık fonksiyonları Şekil 1.7'deki gibidir.



Şekil 1.7. Haar Ölçek Fonksiyonu (a), Haar Dalgacık Fonksiyonu (b)

Haar Dalgacık dönüşümü kullanılarak RGB bir görselin her bir renk kanalı için frekans uzayına geçiş sağlanabilir. Dalgacık dönüşümü sonrasında elde edilen katsayılar, satır ve sütunlara uygulanan yüksek ve alçak geçirgen filtreler sonucunda meydana gelir [22]. Ters dalgacık dönüşümü sonrasında ise zaman uzayına geçiş sağlanarak görsel tekrar original haline getirilebilir. Frekans uzayına geçirilmiş bir görüntüde dalgacık dönüşümünden elde edilen katsayılar kullanılarak dinamik mesafe sıkıştırılması gibi iyileştirmeler yapılabilir [23].

1.4. Veri Yapılarında Sıralama

1.4.1. Veri yapılarında yığın sıralaması: LIFO

Homojen elemanlardan oluşan, sadece tepe kısmında ekleme ve çıkarma gibi işlemler yapılabilen veri yapılarına yığın adı verilir. Örneğin, bir kafeteryadaki tepsilerin oluşturduğu yığın içindeki 2. tepsiye ulaşılabilmesi için, önce üzerinde bulunan 1. tepsinin alınması gerekmektedir (Şekil 1.8).

Bu veri yapısına LIFO (Last In First Out) adı verilerek, son eklenen verinin ilk olarak işleme sokulması gerektiğini nitelemektedir.

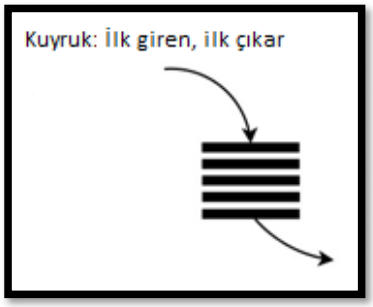
Şekil 1.8. LIFO Örneği: Tepsi Yığını

1.4.2. Veri yapılarında kuyruk sıralaması: FIFO

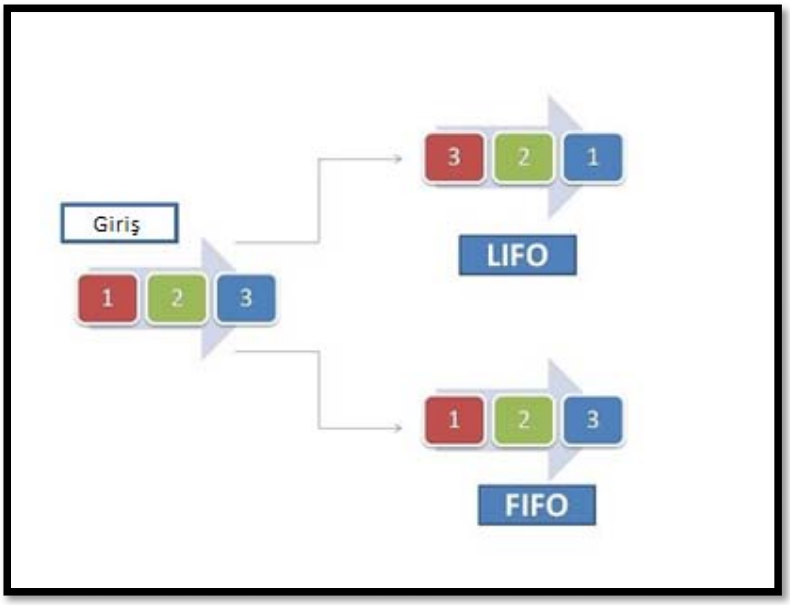
Aynı tipteki verilerin son kısmında ekleme; tepe kısmında ise çıkarma işlemleri yapılabilen veri yapılarına kuyruk adı verilir. İnsanların gerçek hayatta oluşturdukları kuyruk yapısının özelliklerini aynı şekilde taşır. Örneğin, kafeteryadaki yemek almak için sıraya giren insanların oluşturduğu kuyruk incelendiğinde, sıraya ilk giren kişi ilk olarak yemeğini alacaktır. Sıraya son giren kişi ise en arkada sırasını bekleyecektir (Şekil 1.9).

Bu şekilde çalışan veri yapılarına FIFO (First In Last Out) adı verilerek, ilk eklenen verinin ilk olarak işleme sokulması gerektiğini nitelemektedir [24].

LIFO ve FIFO sıralamaları için algoritma karşılaştırması Şekil 1.10'da gösterilmiştir.



Şekil 1.9. FIFO Örneği: Kasa Sırası



Şekil 1.10. LIFO ve FIFO Algoritmaları Karşılaştırması

Görüntü gibi büyük verilerin şifrelenmesinde kesintisiz (stream) veya blok (block) algoritmaları kullanılarak şifreleme yapılmaktadır. Bu şifreleme algoritmaları kullanılarak şifrelenmiş görüntünün tekrar eski orijinal haline döndürülebilmesi için gereken gizli anahtar elde edilmektedir. Gizli anahtar simetrik veya asimetrik biçimde oluşturulabilir. Simetrik biçimde oluşturulmuş bir gizli anahtar, hem gönderici hem de alıcı tarafından aynı şekilde kullanılarak şifrenin çözülmesi sağlanabilir. Asimetrik biçimde oluşturulmuş bir genel anahtar ile herkes şifreleme işlemini gerçekleştirebilir fakat şifre yalnızca gizli anahtar ile çözülebilir [2, 25]. Simetrik olarak şifrelenmiş ve hem alıcı hem de göndericinin elinde bulunan gizli anahtarın, LIFO ve FIFO algoritmaları kullanılarak görüntünün şifresinin daha zor bir biçimde çözülmesi ile güvenlik seviyesi artırılmıştır.

2. METOTLAR

Şifreleme ve şifre çözme işlemleri, farklı renk yoğunluklarına ve boyutlara sahip 4 farklı RGB görüntü üzerinde gerçekleştirilmiştir. Çözünürlükleri 720x720 olan uçak, 330x330 olan penguen, 256x256 olan insan ve 478x478 olan kuş temalı RGB görseller kullanılmıştır. Tüm işlemler, Mathworks firmasının (Mathworks, MA, USA) MATLAB R2015a programı kullanılarak gerçekleştirilmiştir. Aşağıda belirtilen adımlar, MATLAB programında yazılan şifreleme algoritması üzerinde gerçekleştirilen işlemlere göre açıklanmıştır.

2.1. Basit Şifreleme-Şifre Çözme

2.1.1. Orijinal görüntünün rastgele permütasyon ile şifrelenmesi

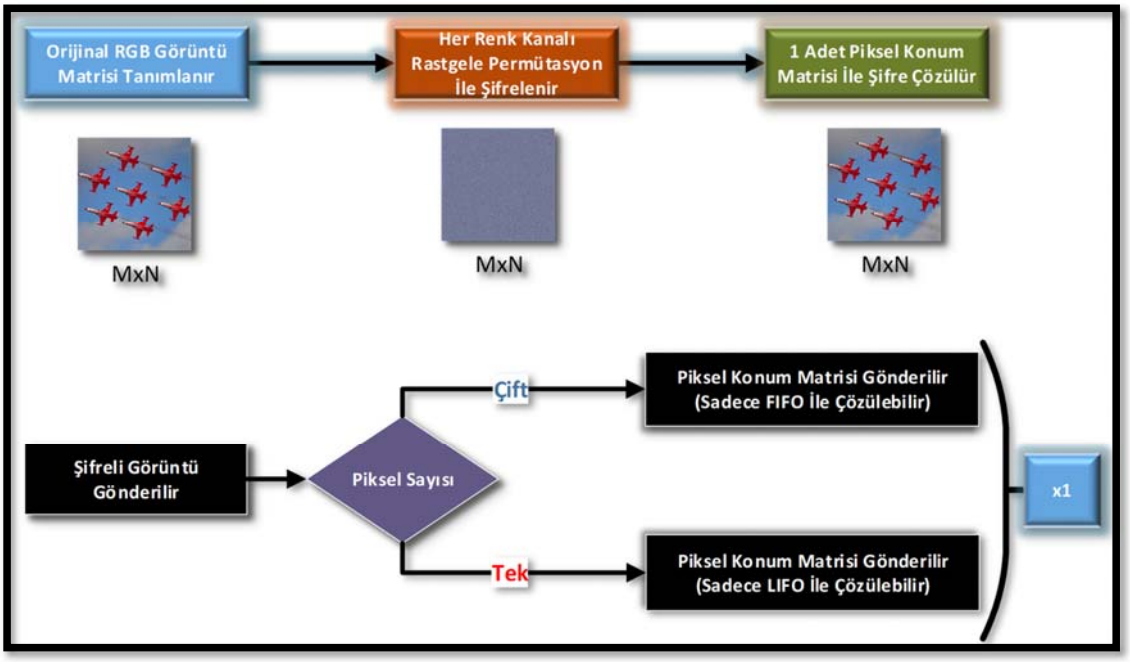
Orijinal olarak elde edilen MxN boyutundaki RGB bir görüntü, zaman uzayında rastgele permütasyon işlemi ile [3] aşağıda belirtilen adımlar doğrultusunda şifrelenmiştir.

1. MxN boyutundaki RGB görüntü matrisi tanımlanır.
2. Görüntünün piksel miktarı (MxN) boyutunda bir satır vektörü oluşturulur.
3. Oluşturulan satır vektörünün indis sıralamalarına rastgele bir permütasyon işlemi uygulanır. Piksel sayısı kadar olan matris elemanlarının rastgele bir biçimde vektörel sıralaması gerçekleştirilir. Bu vektör, şifreleme ve şifre çözme için kullanılacak sıralama anahtarıdır.
4. Matrisi tanımlanan görüntünün R, G ve B kanalları ayrıştırılır.
5. Her bir kanalın elemanları, şifreleme vektörünün sıralamasına uygun şekilde tekrar sıralanır. (İki sütuna ve görüntünün piksel miktarı kadar satır sayısına sahip sıfır matrisi oluşturulur. İlk sütun orijinal görselin piksellerinin konum vektörü, ikinci sütun ise şifreli görselin piksellerinin konumlarını içeren vektör olarak tanımlanır. Şifreli görselin konumları, orijinal görselin piksel konumlarına göre sıralanarak rastgele permütasyon ile şifrelenmiş görüntünün şifresinin çözülmesi için gereken anahtar elde edilir.)
6. Her bir kanal tekrar MxN boyutunda 2-D (iki boyutlu) matris haline getirilerek birleştirilir. Bu şekilde şifrelenmiş görüntü oluşur.

2.1.2. Rastgele permütasyon ile şifrelenmiş orijinal görüntünün aktarımı ve şifrenin çözülmesi

Zaman uzayında şifrelenmiş görüntünün alıcıya gönderilmesi için aşağıdaki adımlar uygulanmıştır.

1. Şifrelenmiş görüntü ile birlikte;
   a. Eğer şifreli görüntünün piksel miktarı çift ise;
      - Şifrenin çözülmesinde kullanılacak vektör, N sütun sayısı olacak şekilde MxN boyutunda bir matrise dönüştürülür. Bu matris anahtar olarak yollanır. Alıcı bu matrisi FIFO algoritması kullanarak tekrar eski vektör haline dönüştürür.
   b. Eğer piksel miktarı tek ise;
      - Bu satır vektörü, N sütun sayısı olacak şekilde MxN boyutunda bir matrise dönüştürülür ve satır konumları ters çevrilir. Bu matris anahtar olarak yollanır. Alıcı bu matrisi LIFO algoritması kullanarak tekrar eski vektör haline dönüştürür.
2. Şifreli görüntünün piksel miktarı boyutunda satır sayısı kadar 2 sütunlu, (MxN)x2 boyutunda bir matris oluşturulur. Bu matrisin ilk sütunu 1'den MxN'e kadar olacak şekilde numaralandırılır. Diğer sütuna ise, şifrenin çözülmesinde kullanılacak vektör indisleri sırasıyla yerleştirilir. İkinci sütun, birinci sütun değerlerine göre sıralanarak piksel konumları çözülür.
3. Şifreli görüntünün R, G ve B kanalları ayrıştırılarak, piksel konumlarını belirten iki sütunlu matris sayesinde şifrelenmemiş orijinal görüntü sıralaması tekrar elde edilir.
4. Her bir kanal tekrar MxN boyutunda 2-D matris haline getirilerek birleştirilir. Bu şekilde şifrelenmiş görüntünün şifresi çözülmüş oluşur.



Şekil 2.1. Rastgele Permütasyon ile Şifreleme-Şifre Çözme

2.2. Dalgacık ile Şifreleme-Şifre Çözme

2.2.1. Ayrık dalgacık dönüşümü ve rastgele permütasyon uygulanarak orijinal görüntünün şifrelenmesi

Daha önce rastgele permütasyon uygulanarak şifrelenen görüntüler bu aşamada, sırasıyla önce ayrık dalgacık dönüşümü, daha sonra da rastgele permütasyon işlemi uygulanarak şifrelenmiştir. Şifreleme işlemi aşağıdaki adımlar doğrultusunda gerçekleştirilmiştir.

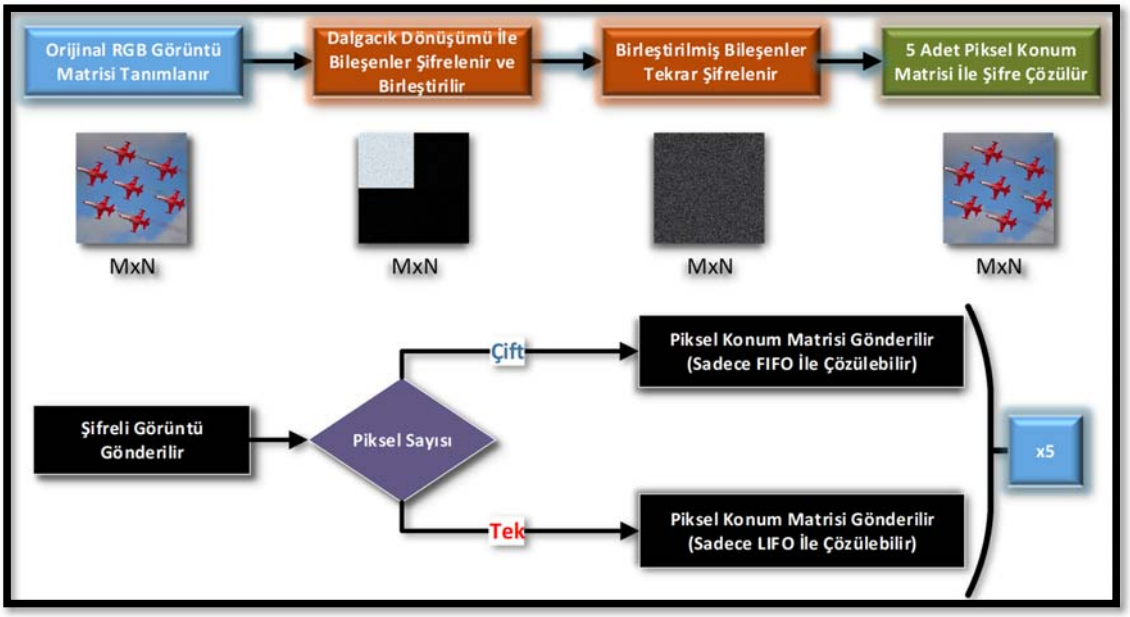
1. MxN boyutundaki RGB görüntü açılır.
2. Görüntünün her bir renk kanalına birinci dereceden Haar ayrık dalgacık dönüşümü uygulanır. Bu şekilde görüntünün yaklaşık (approximate), yatay (horizontal), dikey (vertical) ve köşegen (diagonal) bileşenleri elde edilir. Bu bileşenlerin matris boyutları, orijinal görüntünün matris boyutunun yarısı kadardır (M/2 x N/2).
3. Görüntünün bileşenlerinin piksel miktarları, her bir bileşen için ayrı olacak şekilde (M/2 x N/2) boyutunda 4 adet satır vektörü şeklinde oluşturulur.

4. Oluşturulan satır vektörlerinin indis sıralamalarına rastgele bir permütasyon işlemi uygulanır. Bu vektörler, şifreleme ve şifre çözme için kullanılacak sıralama anahtarlarıdır.
5. Dört bileşenin de R, G ve B kanalları ayrıştırılır. Yatay, dikey ve köşegen bileşenlerin etkisi az olduğundan ihmal edilir. Bu nedenle bu 3 bileşen kendi boyutlarındaki sıfır matrisi ile tanımlanırlar.
6. Her bir kanalın elemanları, şifreleme vektörlerinin sıralamasına uygun şekilde tekrar sıralanır.
7. Her bir kanal tekrar M/2 x N/2 boyutunda 2-D matrisler haline getirilerek birleştirilir. Bu şekilde şifrelenmiş approximation katsayı görüntüsü oluşur.
8. Şifrelenmiş 4 adet görüntü A, H, V, D sıralamasıyla MxN boyutunda tekrar birleştirilir ve bu görüntüye aynı şekilde rastgele permütasyon ile şifreleme işlemi tekrar uygulanır.

2.2.2. Ayrık dalgacık dönüşümü ve rastgele permütasyon ile şifrelenmiş orijinal görüntünün aktarımı ve şifrenin çözülmesi

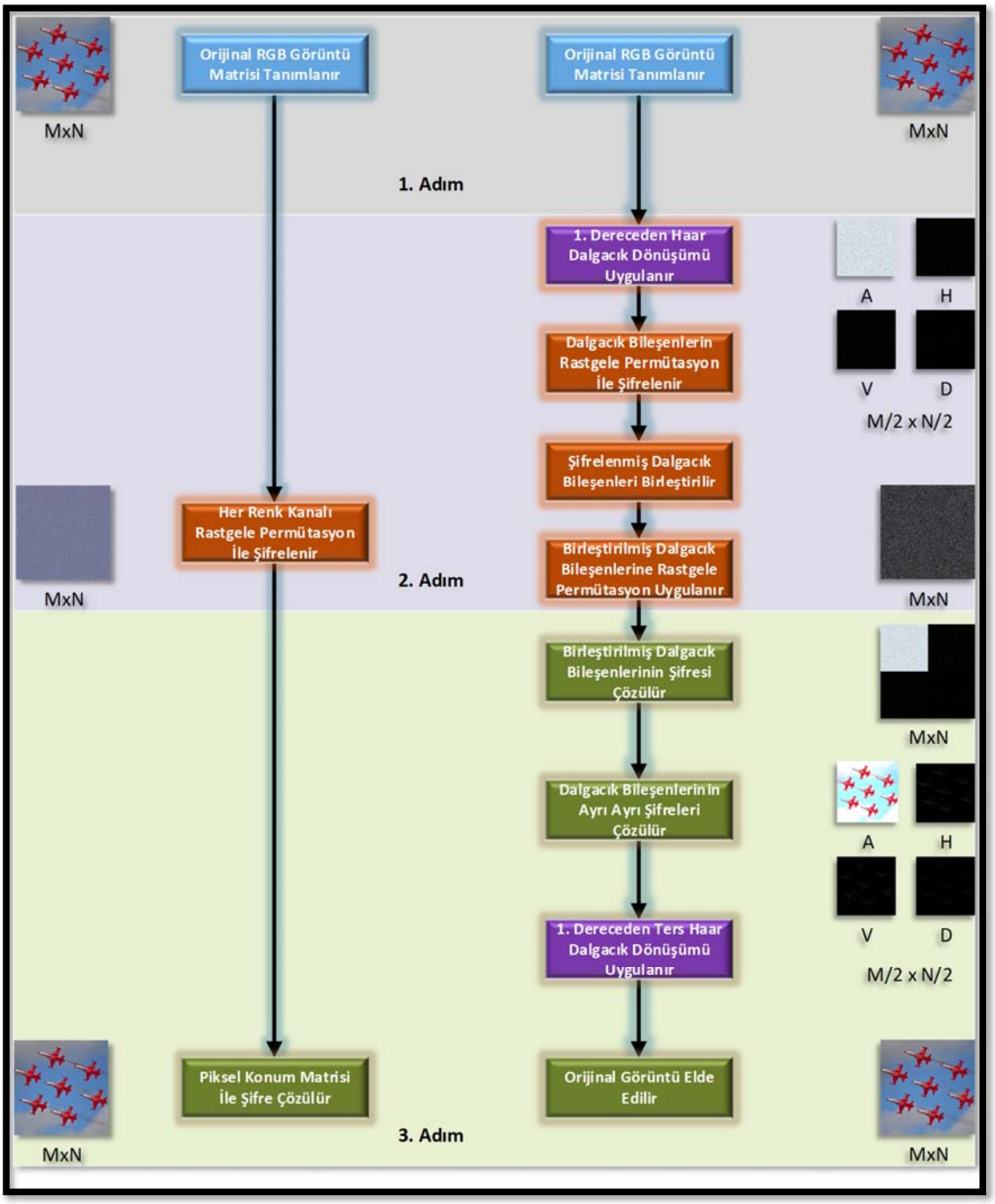
1. Şifrelenmiş görüntü ile birlikte;
   a. Eğer şifreli görüntünün piksel miktarı çift ise;
      - Şifrenin çözülmesinde kullanılacak vektör, N sütun sayısı olacak şekilde MxN boyutunda bir matrise dönüştürülür. Bu matris anahtar olarak yollanır. Alıcı bu matrisi FIFO algoritması kullanarak tekrar eski vektör haline dönüştürür.
   b. Eğer piksel miktarı tek ise;
      - Bu satır vektörü, N sütun sayısı olacak şekilde MxN boyutunda bir matrise dönüştürülür ve satır konumları ters çevrilir. Bu matris anahtar olarak yollanır. Alıcı bu matrisi LIFO algoritması kullanarak tekrar eski vektör haline dönüştürür.
2. Şifreli görüntünün piksel miktarı boyutunda satır sayısı kadar 2 sütunlu, (MxN)x2 boyutunda bir matris oluşturulur. Bu matrisin ilk sütunu 1'den MxN'e kadar olacak şekilde numaralandırılır. Diğer sütuna ise, şifrenin çözülmesinde kullanılacak vektör indisleri sırasıyla yerleştirilir. İkinci sütun, birinci sütun değerlerine göre sıralanarak piksel konumları çözülür.

3. Şifreli görüntünün R, G ve B kanalları ayrıştırılarak, piksel konumlarını belirten iki sütunlu matris sayesinde şifrelenmemiş orijinal görüntü sıralaması tekrar elde edilir.
4. Her bir kanal tekrar MxN boyutunda 2-D matris haline getirilerek birleştirilir. Bu şekilde şifrelenmiş görüntünün şifresi çözülmüş oluşur.
5. Orijinal hale getirilen R, G ve B kanalları birleştirilerek; şifrelenmiş ve birleştirilmiş A, H, V, D (yaklaşık, yatay, dikey, köşegen detay) görüntüsü elde edilir.
6. M/2xN/2 boyutundaki approximation katsayı görüntüsü için olan 2. anahtar da kullanılarak şifre çözülerek orijinal görüntü elde edilir.



Şekil 2.2. Dalgacık Dönüşümü ile Şifreleme-Şifre Çözme

Basit şifreleme-şifre çözme metodu ile birlikte ayrık dalgacık dönüşümü kullanılarak şifreleme-şifre çözme aşamaları Şekil 2.3'de özetlenmiştir.



Şekil 2.3. Rastgele Permütasyon Yöntemi ve Ayrık Dalgacık Dönüşümü ile Şifreleme Basamakları

Satır sayısı M ve sütun sayısı N olan bir görüntünün rastgele permütasyon ile piksel konumlarının yer değiştirilmesi için öncelikle sütun sayısı kadar sütun vektörlerine ayrışım gerçekleştirilir.

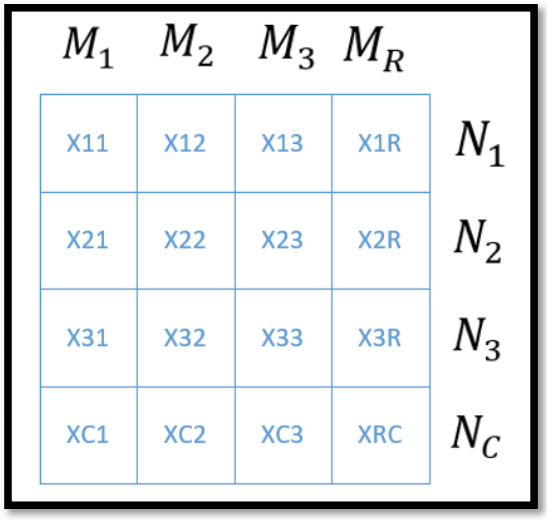
$$N_1 + N_2 + \cdots + N_C = N \tag{2.1}$$

Daha sonrasında ise satır sayısı kadar satır vektörlerine ayrışım uygulanır.

$$M_1 + M_2 + \cdots + M_R = M \tag{2.2}$$

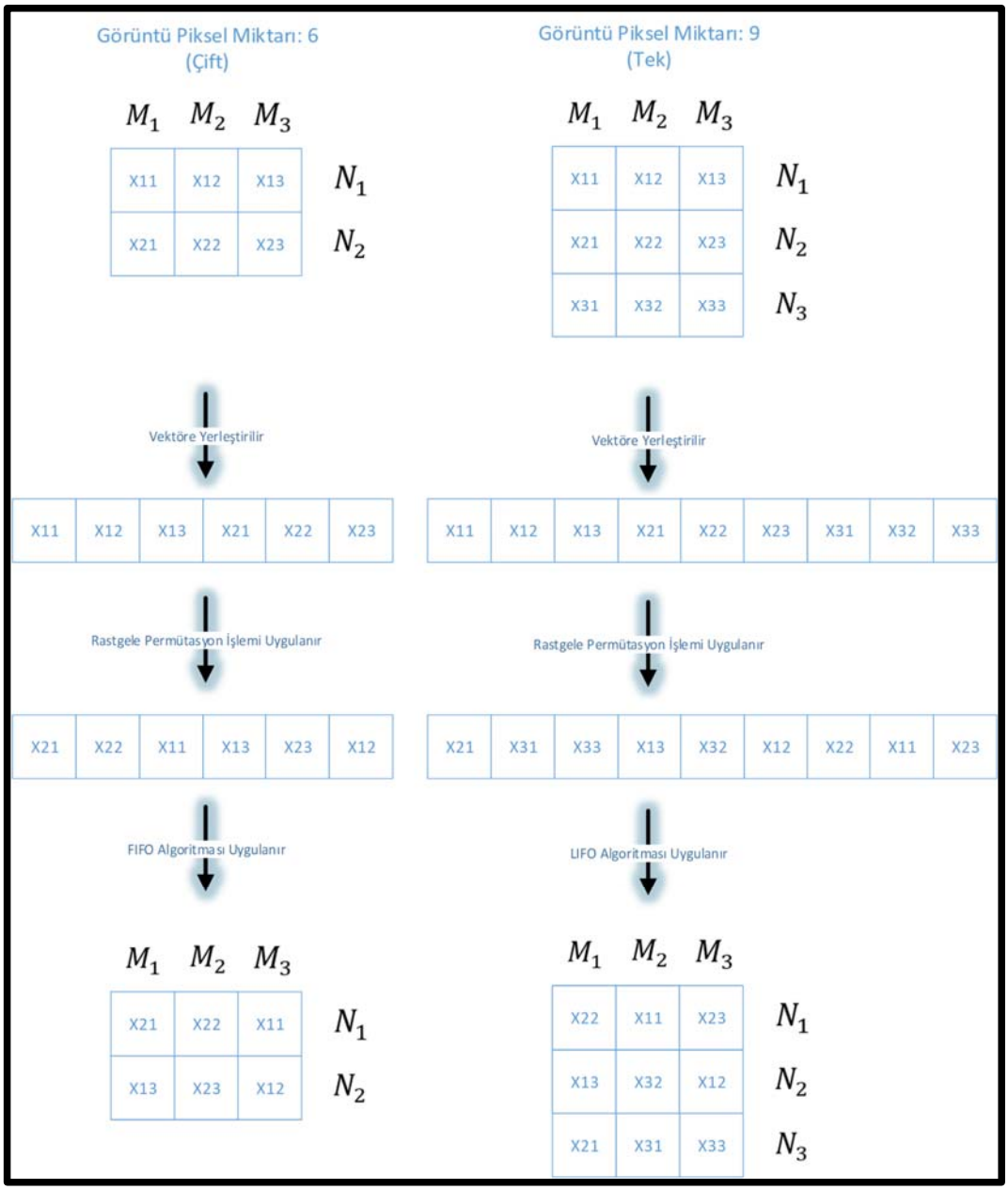
Her bir satır vektörü sütun vektörüne, her bir sütun vektörü ise satır vektörü miktarına bölünerek pikseller tanımlanır (Şekil 2.4). Bu piksellerin tümü tek bir vektöre yerleştirilerek vektör uzunluğundaki rastgele sıralama işlemi gerçekleştirilerek rastgele permütasyon işlemi (2.3) uygulanır.

$$XRC = sort\big(rand(1,n)\big)\,|_{n:vektör\ uzunluğu} \tag{2.3}$$



Şekil 2.4. Görüntünün Piksellere Ayrıştırılması

Her permütasyon işleminden sonra piksel sayısının çift veya tek olmasına göre 1xn boyutundaki satır vektörünün LIFO veya FIFO algoritması kullanılmasıyla tekrar MxN boyutuna doğru sıralama ile dönüştürülmesi gerçekleştirilir (Şekil 2.5).



Şekil 2.5. Görüntünün Piksellere Ayrıştırılması

Alıcıya gönderilen şifreli görüntünün tekrar 1xn boyutundaki vektöre dönüştürülmesi sırasında; şifreleme işleminde uygulanan, piksel miktarına bağlı olarak değişen algoritma ile gerçekleştirilmesi gerekmektedir.

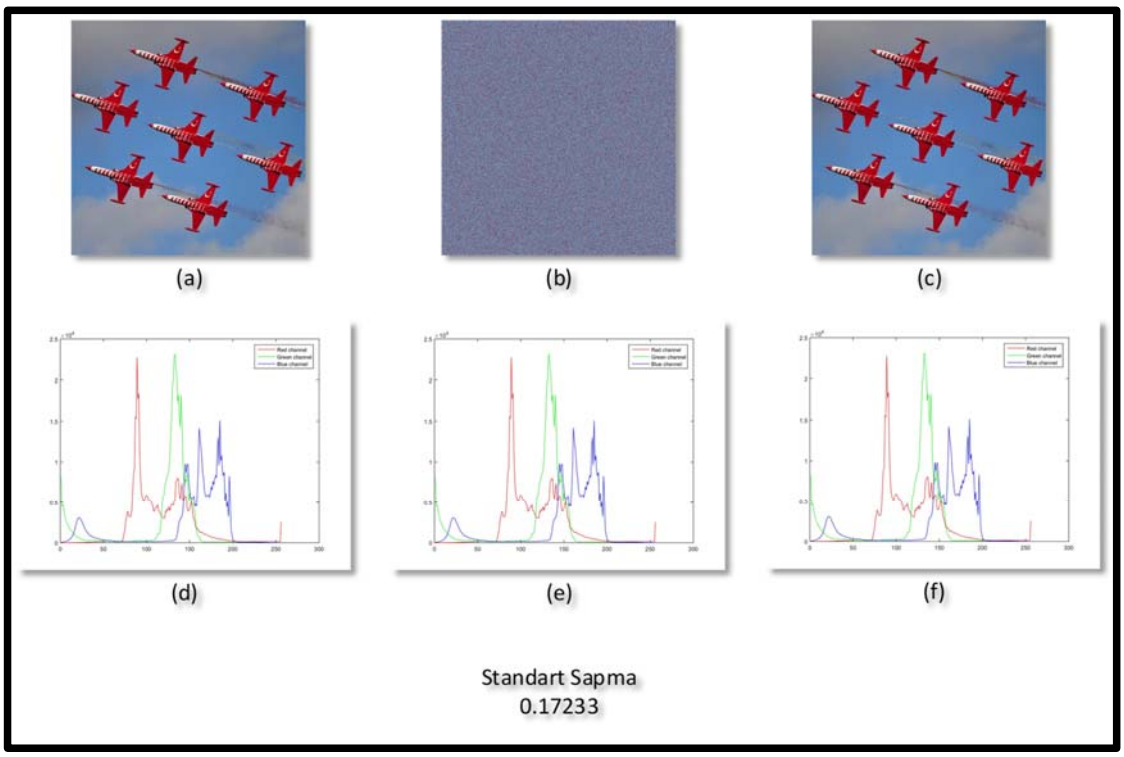
3. BULGULAR

Metotlar başlığı altında bahsedilen işlem adımları, gerçek görüntüler üzerinde Mathworks firmasının (Mathworks, MA, USA) MATLAB R2015a programı kullanılarak gerçekleştirilmiştir (EK 1). Oluşturulan yazılım ile 4 farklı görsel için sadece rastgele permütasyon kullanılarak basit şifreleme-şifre çözme ve her bir dalgacık bileşenine rastgele permütasyon uygulanan dalgacık ile şifreleme-şifre çözme yöntemleriyle sonuçlar elde edilmiştir.

3.1. Basit Şifreleme-Şifre Çözme Uygulaması

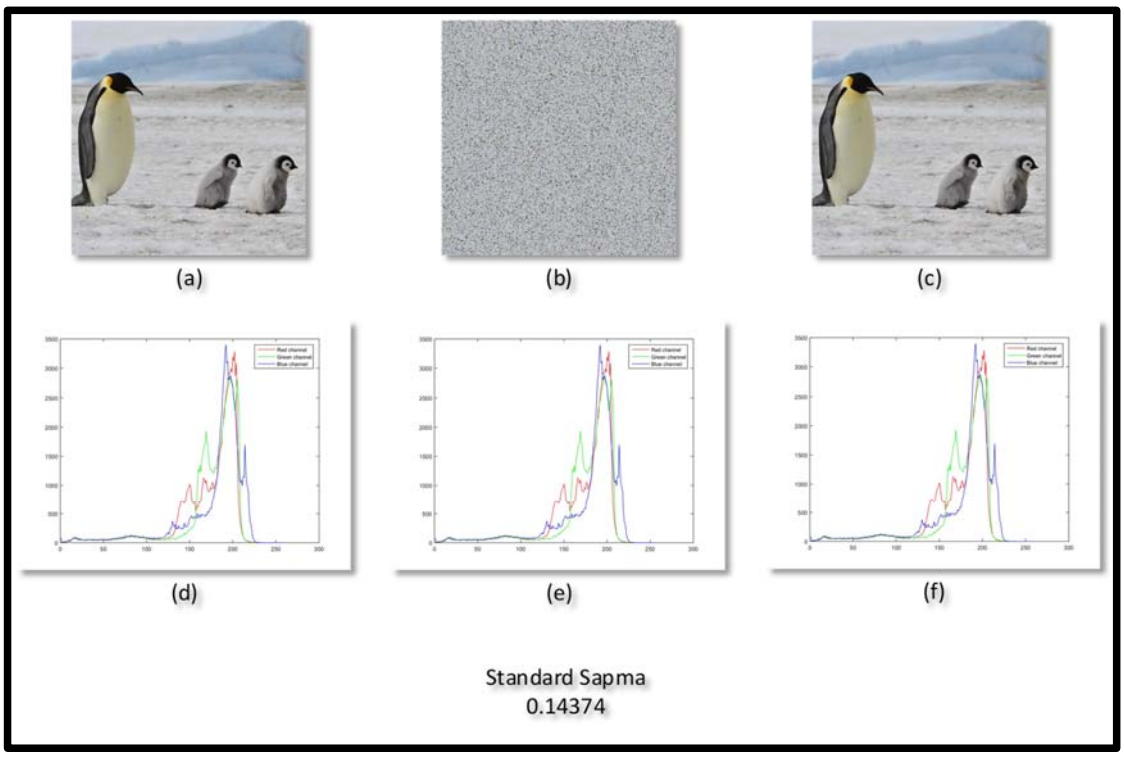
Çözünürlükleri 720x720 olan uçak (Şekil 3.1), 330x330 olan penguen (Şekil 3.2), 256x256 olan insan (Şekil 3.3) ve 478x478 olan kuş (Şekil 3.4) temalı RGB görsellerin, rastgele permütasyon ile şifrelenmiş görüntülere ait histogram dağılımları oluşturulmuştur. Her piksel seviyesinin imgedeki miktarını (frekansını) göstermek için histogram kullanılır [17]. Histogram grafiğindeki yatay eksende pozitif yöne doğru artan veri, ilgili renk kanalının ne kadar parlak olduğunu; dikey eksende pozitif yöne doğru artan veri ise ilgili renk kanalının piksel miktarını göstermektedir. Standart sapma değerleri, şifreleme işlemi sonrasında ortalama değerden ne kadar uzaklaştığını göstermektedir. Standart sapma değeri ne kadar yüksek olursa değerler de ortalama değerden o kadar uzaklaşmış olur.

Orijinal uçak görseli, basit şifreleme yöntemi ile elde edilen şifreli görüntü ve şifresi çözülmüş görüntünün standart sapma değerleri (0.17233) kırmızı, yeşil ve mavi kanalların histogram dağılımlarıyla doğru orantılı bir biçimde birbirine eşit olarak elde edilmiştir (Şekil 3.1).



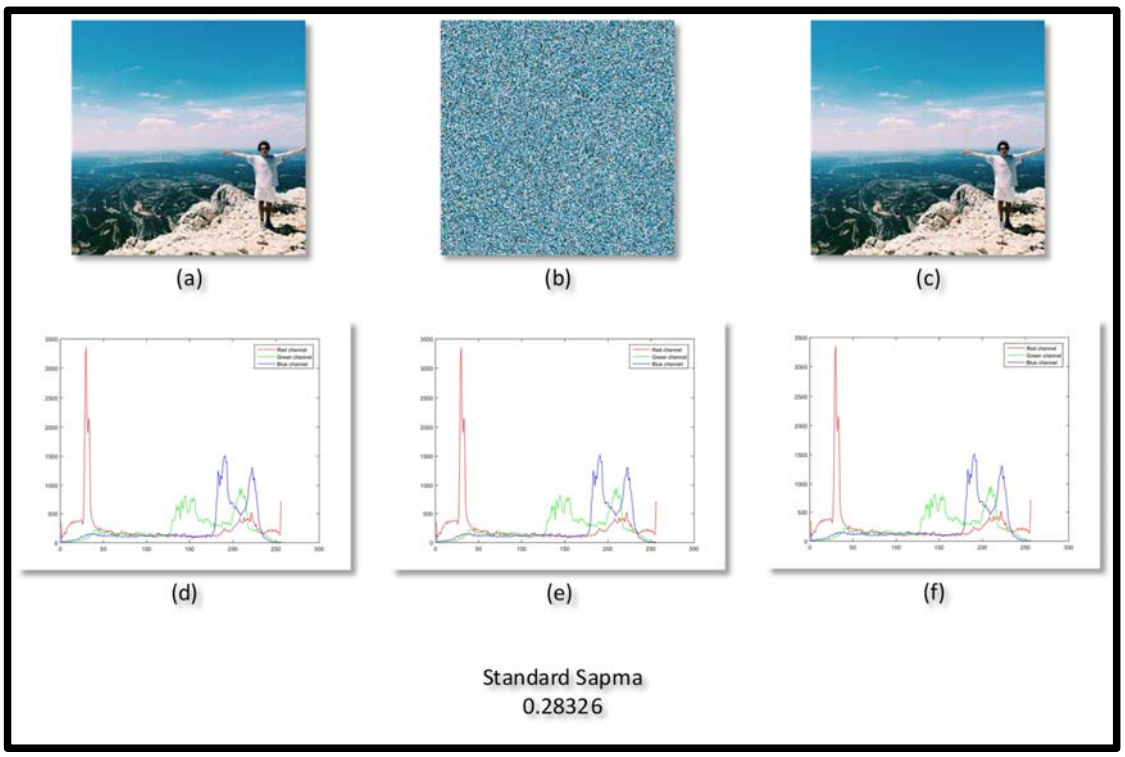
Şekil 3.1. Orijinal Uçak Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)

Orijinal penguen görseli, basit şifreleme yöntemi ile elde edilen şifreli görüntü ve şifresi çözülmüş görüntünün standart sapma değerleri (0.14374) kırmızı, yeşil ve mavi kanalların histogram dağılımlarıyla doğru orantılı bir biçimde birbirine eşit olarak elde edilmiştir (Şekil 3.2). Renk kanallarının orijinal uçak görseline kıyasla birbirlerine yakın ve aynı yoğunlukta olmasından dolayı standart sapma değeri de düşük çıkmıştır.



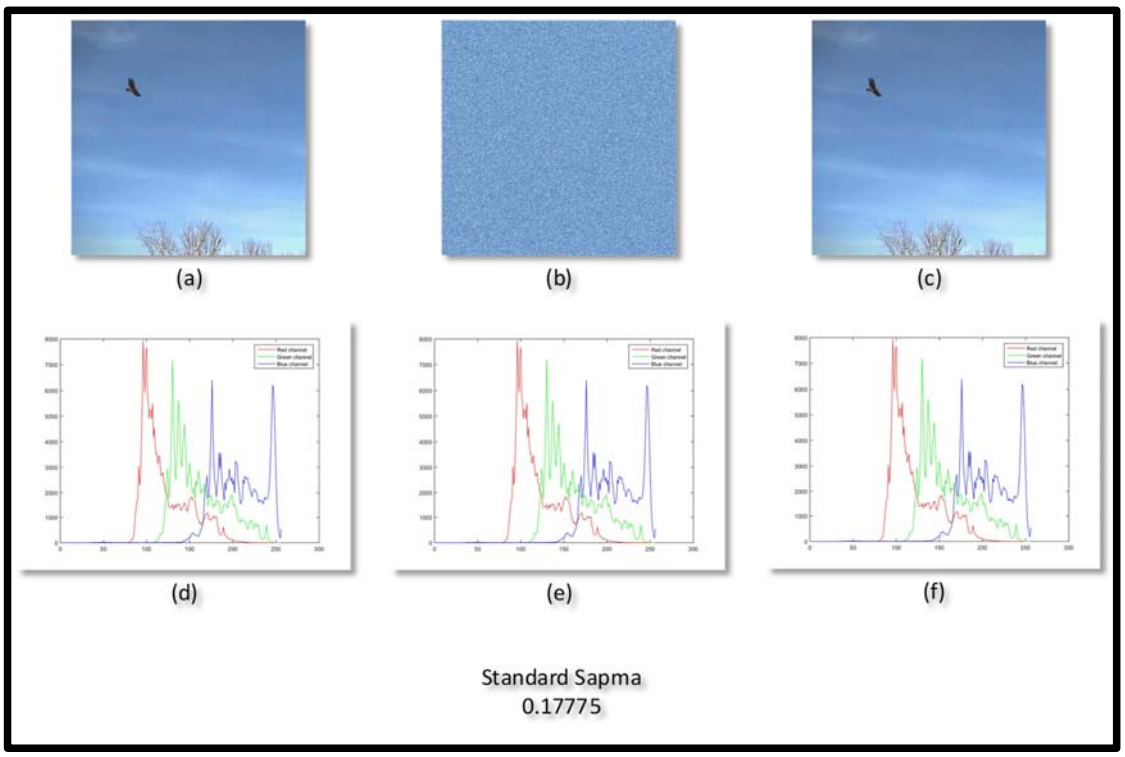
Şekil 3.2. Orijinal Penguen Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)

Orijinal insan görseli, basit şifreleme yöntemi ile elde edilen şifreli görüntü ve şifresi çözülmüş görüntünün standart sapma değerleri (0.28326) kırmızı, yeşil ve mavi kanalların histogram dağılımlarıyla doğru orantılı bir biçimde birbirine eşit olarak elde edilmiştir (Şekil 3.3). Renk kanallarının orijinal uçak ve orijinal penguen görsellerine kıyasla birbirlerine uzak ve daha farklı yoğunluklarda olmasından dolayı standart sapma değeri daha yüksek çıkmıştır.



Şekil 3.3. Orijinal İnsan Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)

Orijinal kuş görseli, basit şifreleme yöntemi ile elde edilen şifreli görüntü ve şifresi çözülmüş görüntünün standart sapma değerleri (0.17775) kırmızı, yeşil ve mavi kanalların histogram dağılımlarıyla doğru orantılı bir biçimde birbirine eşit olarak elde edilmiştir (Şekil 3.4). Renk kanallarının, orijinal uçak görselindeki gibi yoğunluklara sahip olmasından dolayı standart sapma değeri de benzer çıkmıştır.



Şekil 3.4. Orijinal Kuş Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)

Elde edilen sonuçlar renkli bir görüntünün (Şekil 3.1a, Şekil 3.2a, Şekil 3.3a, Şekil 3.4a), piksel konumlarının zaman uzayında yer değiştirilmesiyle anlamsız bir görüntü oluşturduğunu göstermiştir (Şekil 3.1b, Şekil 3.2b, Şekil 3.3b, Şekil 3.4b).

Orijinal anlamlı görüntülerin kırmızı (R), yeşil (G) ve mavi (B) kanallarına uygulanan rastgele ~~bir~~ permütasyon ~~yöntemi~~ işlemi ile ~~şifreleme işlemi gerçekleştirilmiştir~~ şifrelenmiş görüntüler elde edilmiştir. Bu işlem sonucunda hangi piksel verisinin nereye taşınacağı bilgisini içeren bir anahtar vektörü ile şifrelenmiş görüntünün tekrar eski haline getirilmesi sağlanmıştır (Şekil 3.1c, Şekil 3.2c, Şekil 3.3c, Şekil 3.4c).

Zaman uzayında yapılan bu şifreleme yöntemi sonucunda orijinal, şifreli ve şifresi çözülmüş her dört görüntü için histogram verileri incelenmiş, buna göre orijinal, şifreli ve şifresi çözülmüş görüntülere ait histogramların aynı olduğu tespit edilmiştir (Şekil 3.1d, Şekil 3.1e, Şekil 3.1f, Şekil 3.2d, Şekil 3.2e, Şekil 3.2f, Şekil 3.3d, Şekil 3.3e, Şekil 3.3f, Şekil 3.4d, Şekil 3.4e, Şekil 3.4f). Yazılım uygulaması sürecinde renk kanallarında herhangi

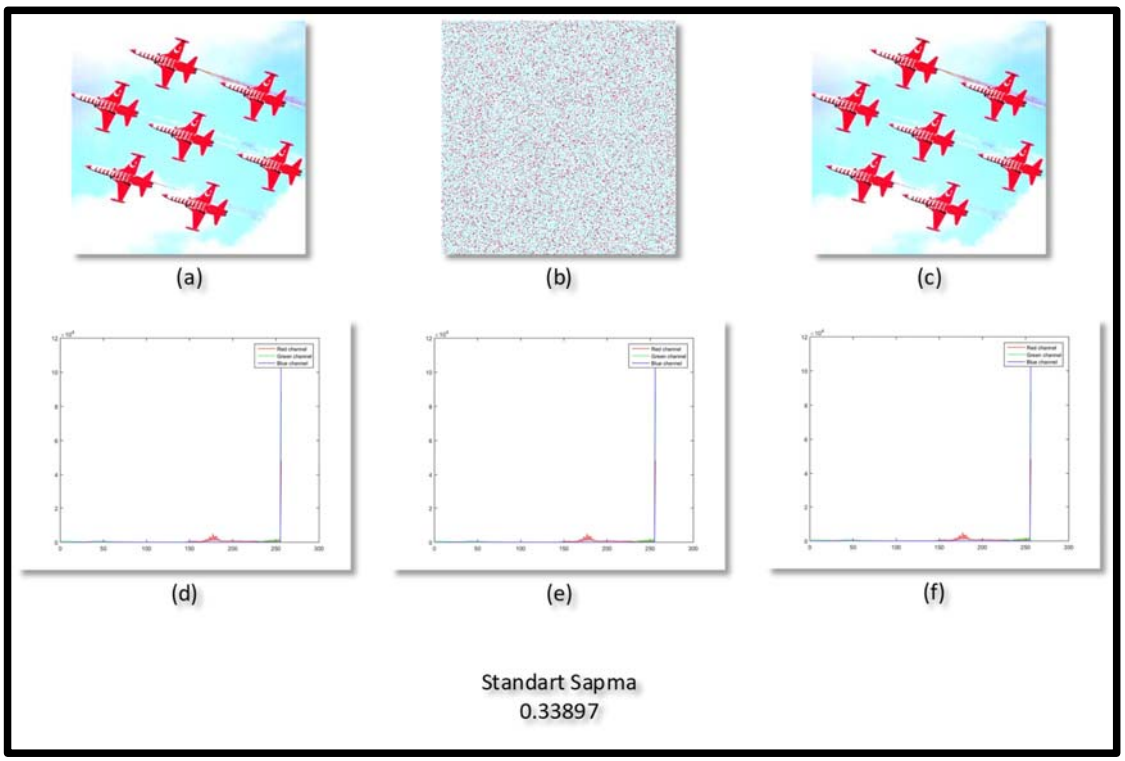
bir değişiklik yapılmaması ve piksellerde sadece yer değiştirme işlemi uygulanmış olması sonucu renk kanallarında bir yoğunluk değişimi de görülmemiştir. Sonuç olarak görsel üzerinde şifreleme kaynaklı herhangi bir veri kaybı gerçekleşmemiştir.

Bu sonuçlar ışığında, Şekil 3.1, Şekil 3.2, Şekil 3.3, Şekil 3.4'deki şifreleme yönteminde, şifrelenmiş görüntülerin orijinal haline geri döndürülebilmeleri için oluşturulmuş piksel sıralama vektörüne bakarak şifrenin kolaylıkla çözülmesi mümkündür.

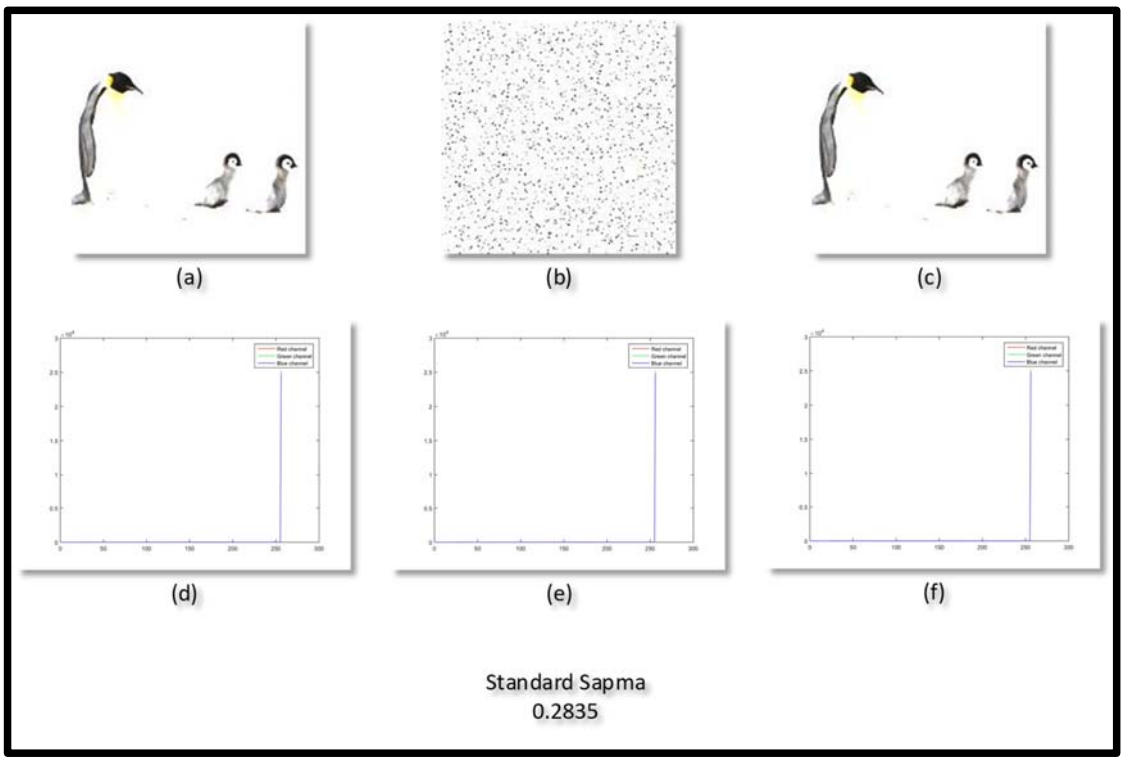
3.2. Dalgacık ile Şifreleme-Şifre Çözme Uygulaması

Şifrelenmiş bir görüntünün, şifresinin kolay bir şekilde çözülememesi için, frekans uzayı gibi zaman uzayından farklı bir ortamda işlem yapılması gerekmektedir. Zaman uzayındaki şifreleme işlemine göre frekans uzayı, şifrelenmiş görüntü için oluşturulacak çözüm anahtarının sadece zamana bağlı olacak şekilde üretilmesinin önüne geçer. Böylece gizlilik düzeyi de artacaktır [26].

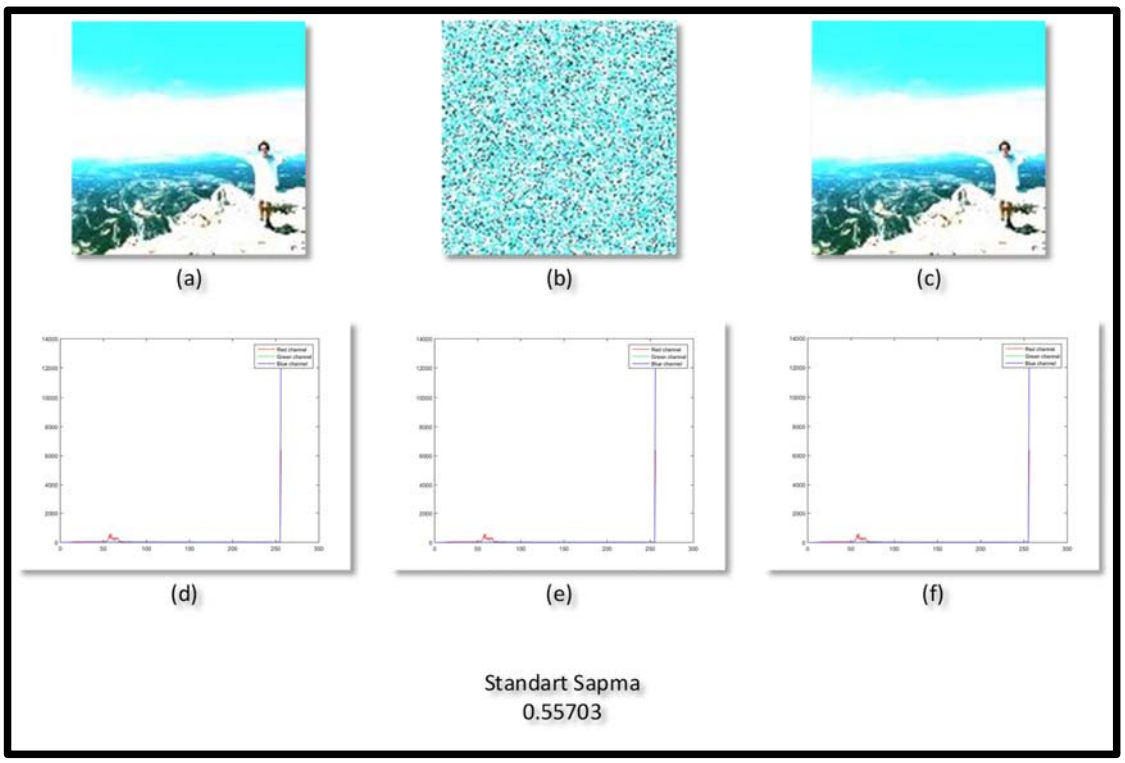
Birinci dereceden Haar ayrık dalgacık dönüşümü uygulanan bir görselin yaklaşık, dikey, yatay ve köşegen görüntü detayları olmak üzere 4 farklı bileşenine ulaşılabilmektedir. Bu bileşenlerin her biri, orijinal görselin 1/4'ü büyüklüğüne sahiplerdir. Orijinal çözünürlükleri 720x720 olan uçak (Şekil 3.5, Şekil 3.6, Şekil 3.7, Şekil 3.8), 330x330 olan penguen (Şekil 3.9, Şekil 3.10, Şekil 3.11, Şekil 3.12), 256x256 olan insan (Şekil 3.13, Şekil 3.14, Şekil 3.15, Şekil 3.16) ve 478x478 olan kuş (Şekil 3.17, Şekil 3.18, Şekil 3.19, Şekil 3.20) temalı RGB görsellerin, rastgele permütasyon ile şifrelenmiş approximation detaylarına ait histogram dağılımları oluşturulmuştur. MxN boyutundaki orijinal görselin 1/4'ü büyüklüğünde olan yaklaşık bileşeni, hem orijinal görselden farklı bir renk yoğunluğuna, hem de büyüklüğü neticesinde orijinal görselin 1/4'ü kadar piksel miktarına sahiptir. Farklı boyutlardaki bu görsellere 1. dereceden ayrık dalgacık dönüşümü uygulandığında elde edilen (M/2) x (N/2) boyutlarındaki yaklaşık (approximation) katsayı görselleri ve histogramları Şekil 3.5, Şekil 3.6, Şekil 3.7, Şekil 3.8' de verilmiştir.



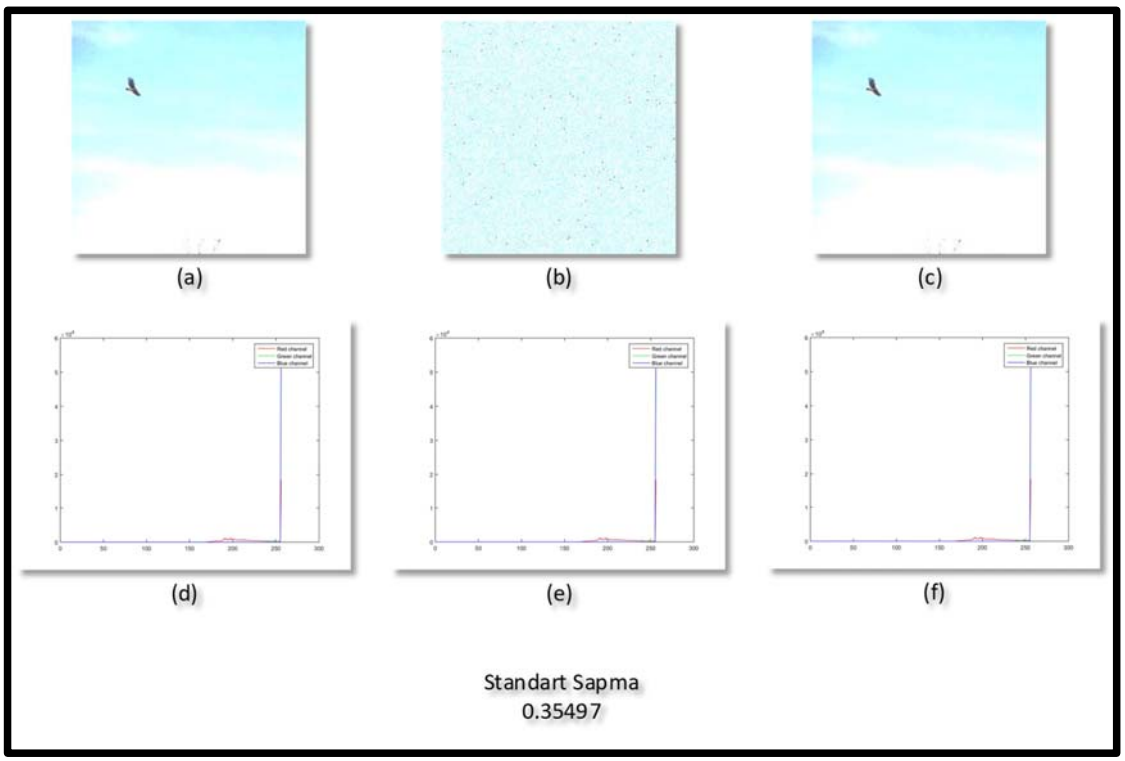
Şekil 3.5. Yaklaşık Uçak Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)



Şekil 3.6. Yaklaşık Penguen Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)

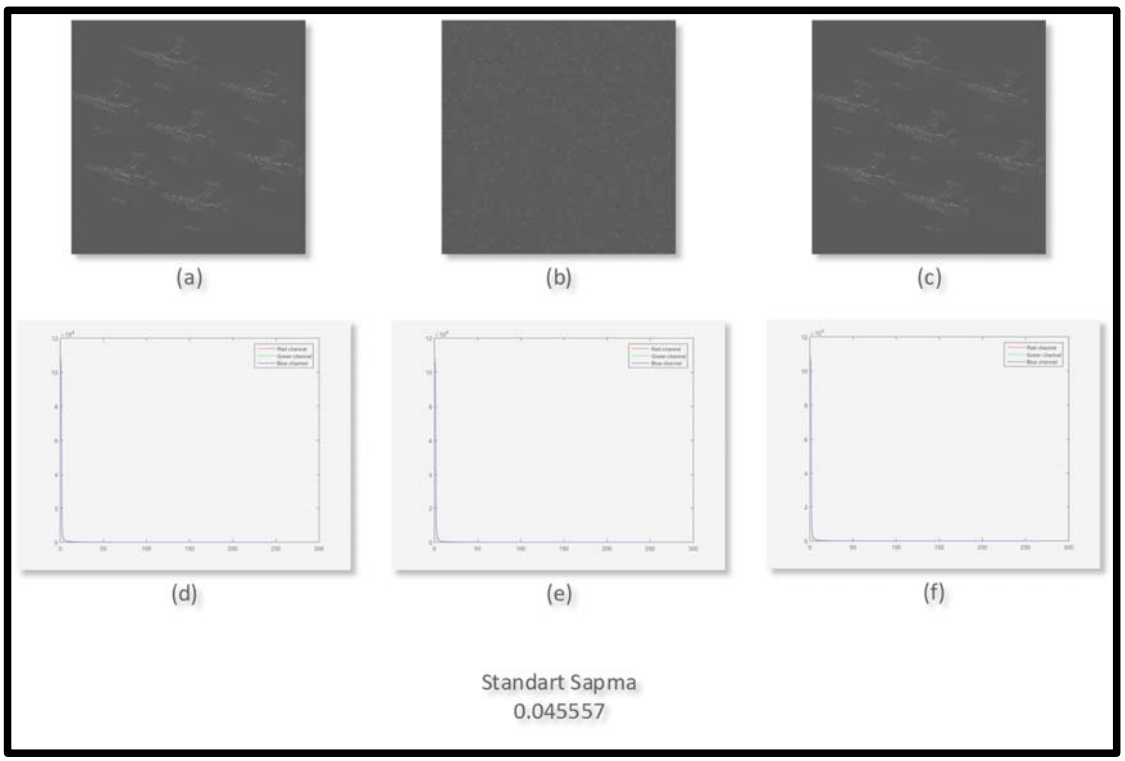


Şekil 3.7. Yaklaşık İnsan Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)

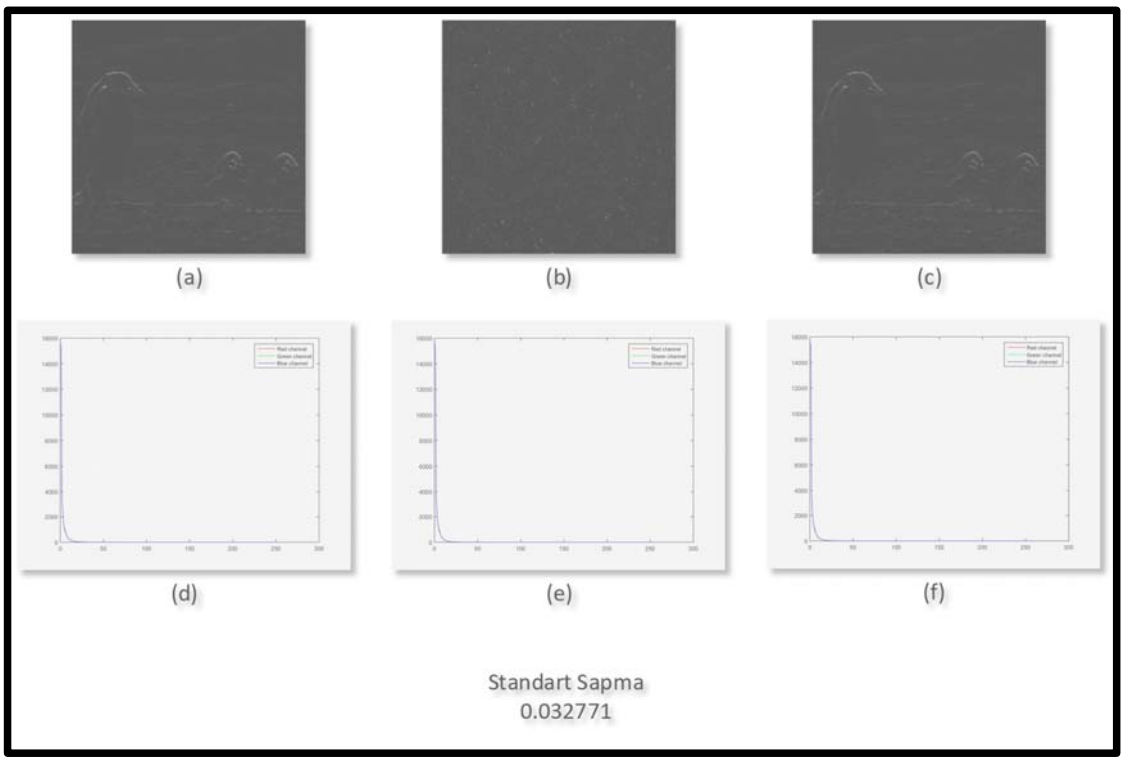


Şekil 3.8. Yaklaşık Kuş Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)

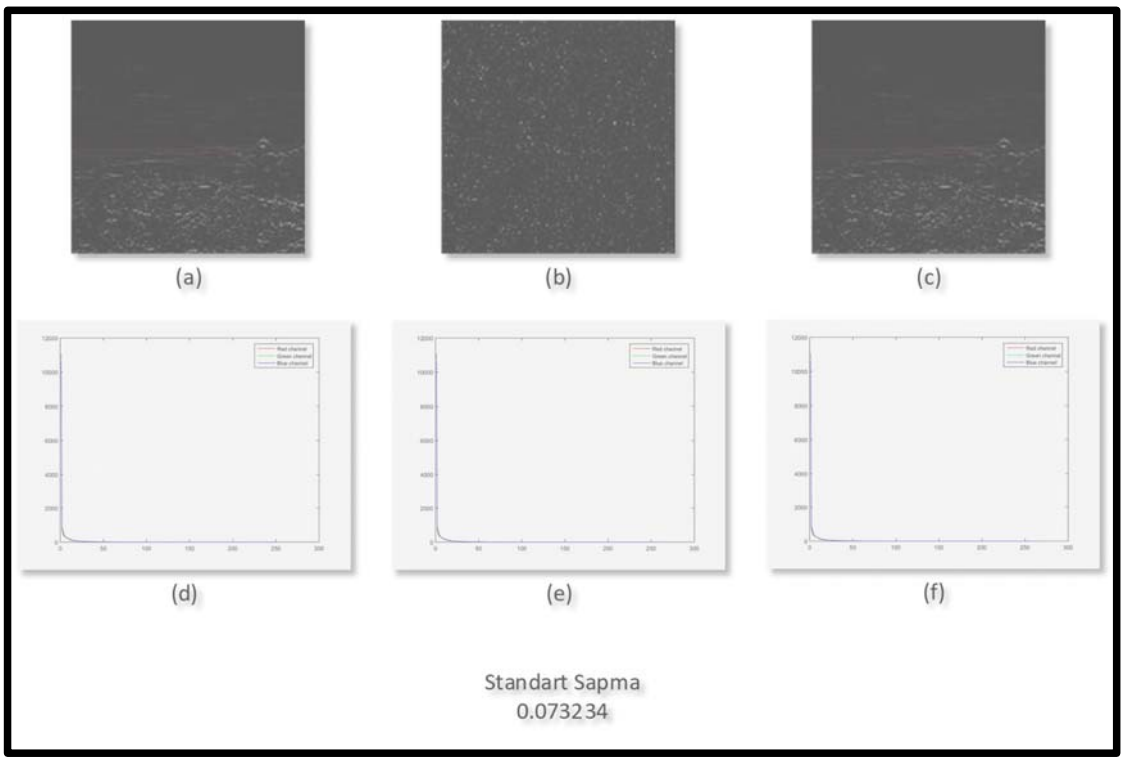
Yukarıda bahsi geçen, ayrık dalgacık dönüşümü sonrasında elde edilen ve yaklaşık görüntü detayları incelenmiş dört görsele ait diğer üç bileşen (yatay, dikey ve köşegen) görüntüleri de rastgele permütasyon ile şifrelenerek histogramları incelenmiştir, sonuçlar Şekil 3.9 – 3.20'de sunulmuştur.



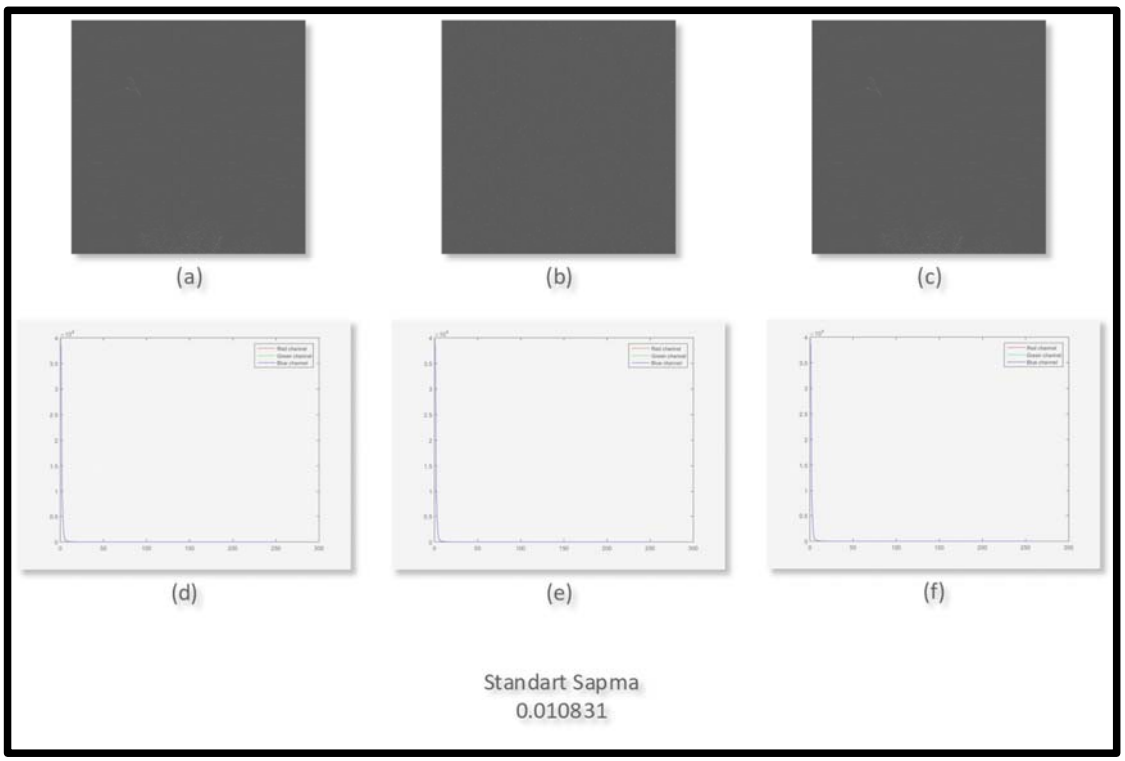
Şekil 3.9. Yatay Uçak Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)



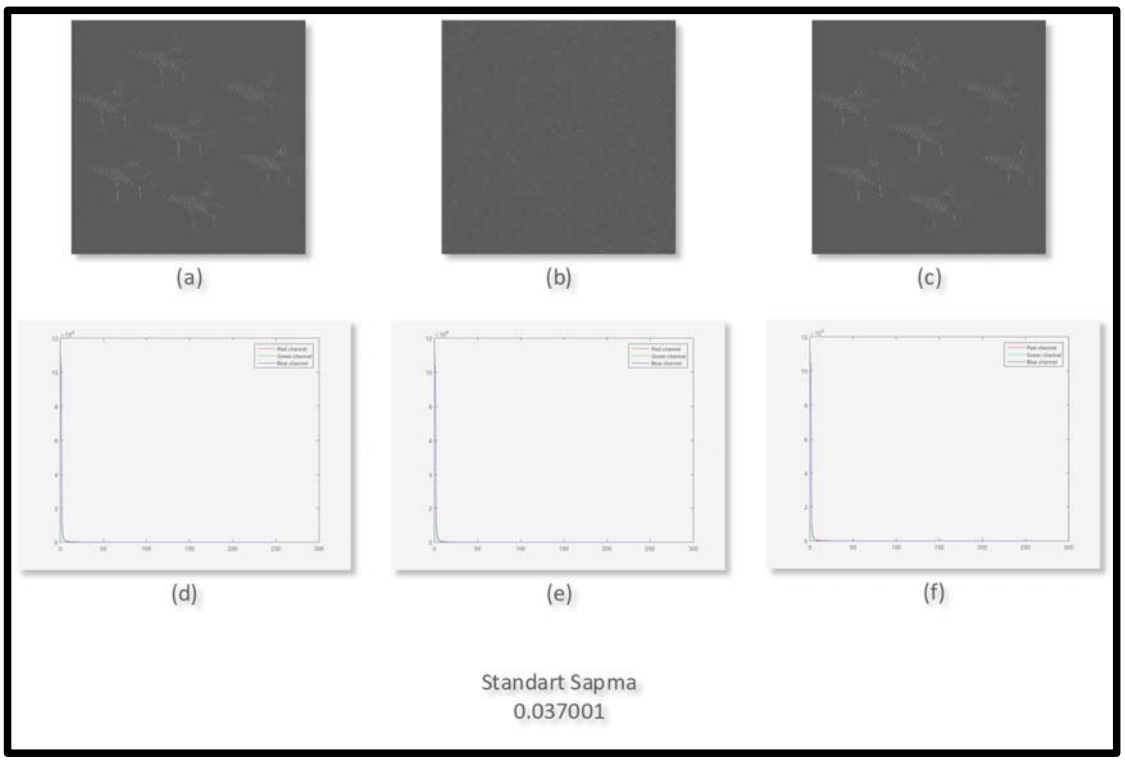
Şekil 3.10. Yatay Penguen Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)



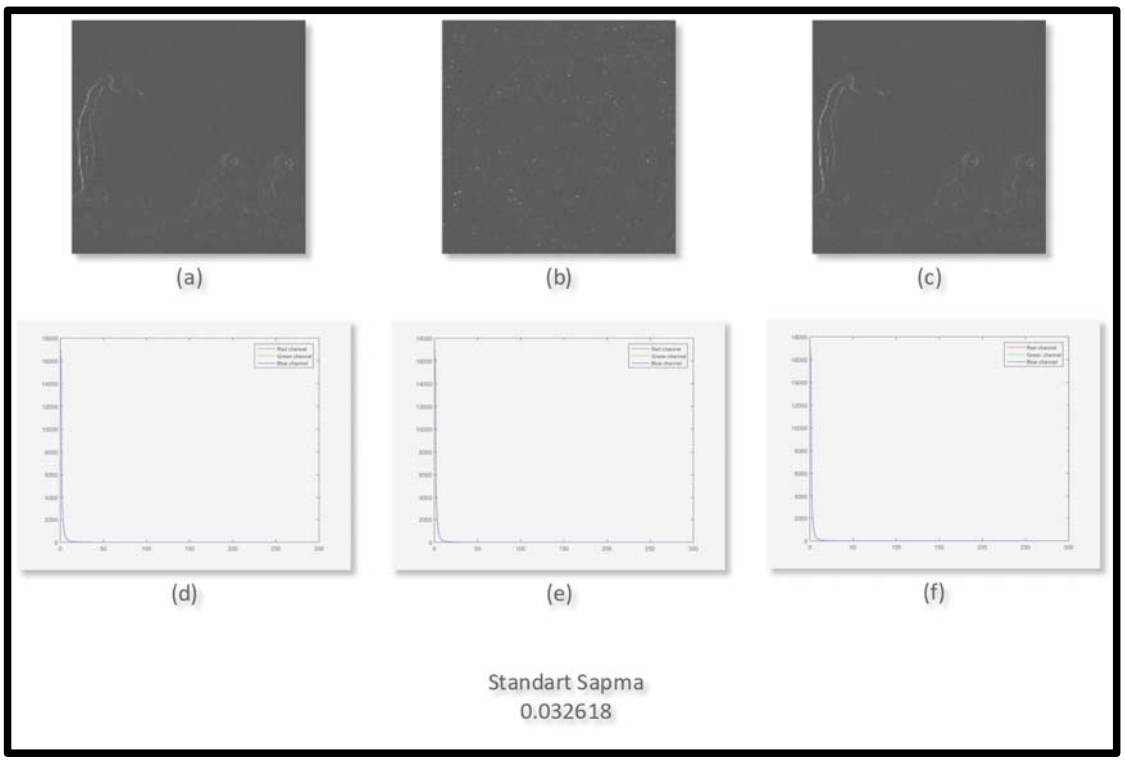
Şekil 3.11. Yatay İnsan Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)



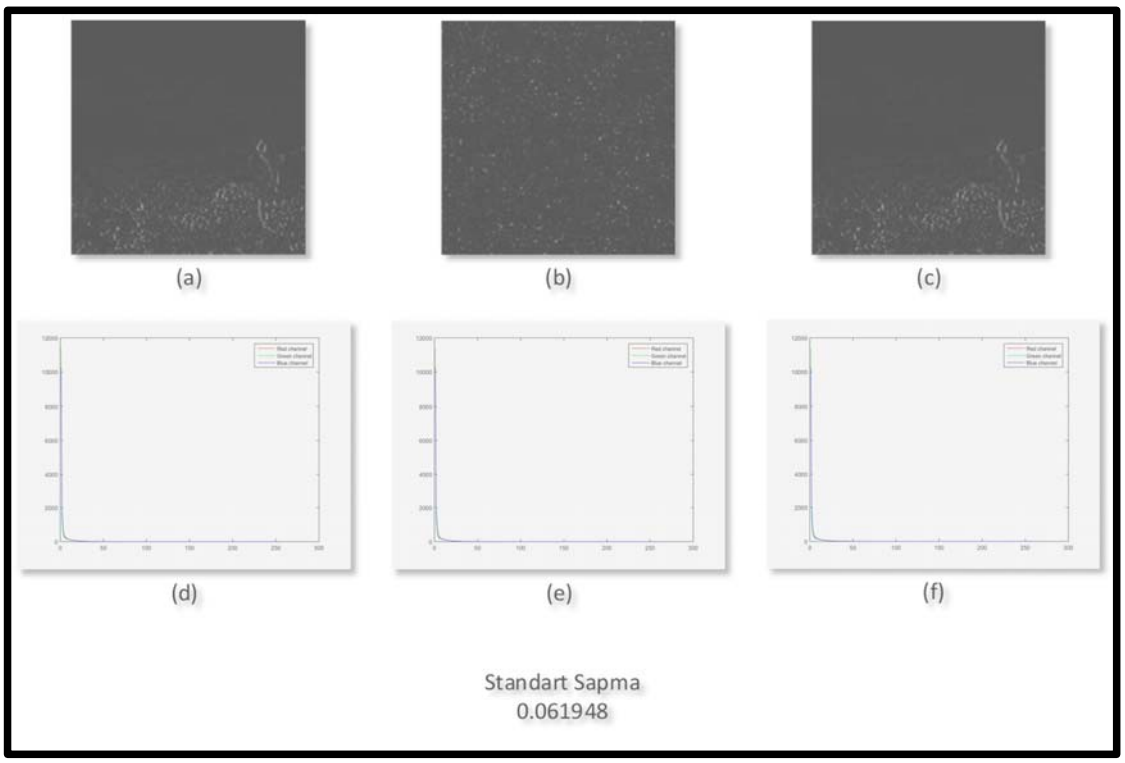
Şekil 3.12. Yatay Kuş Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)



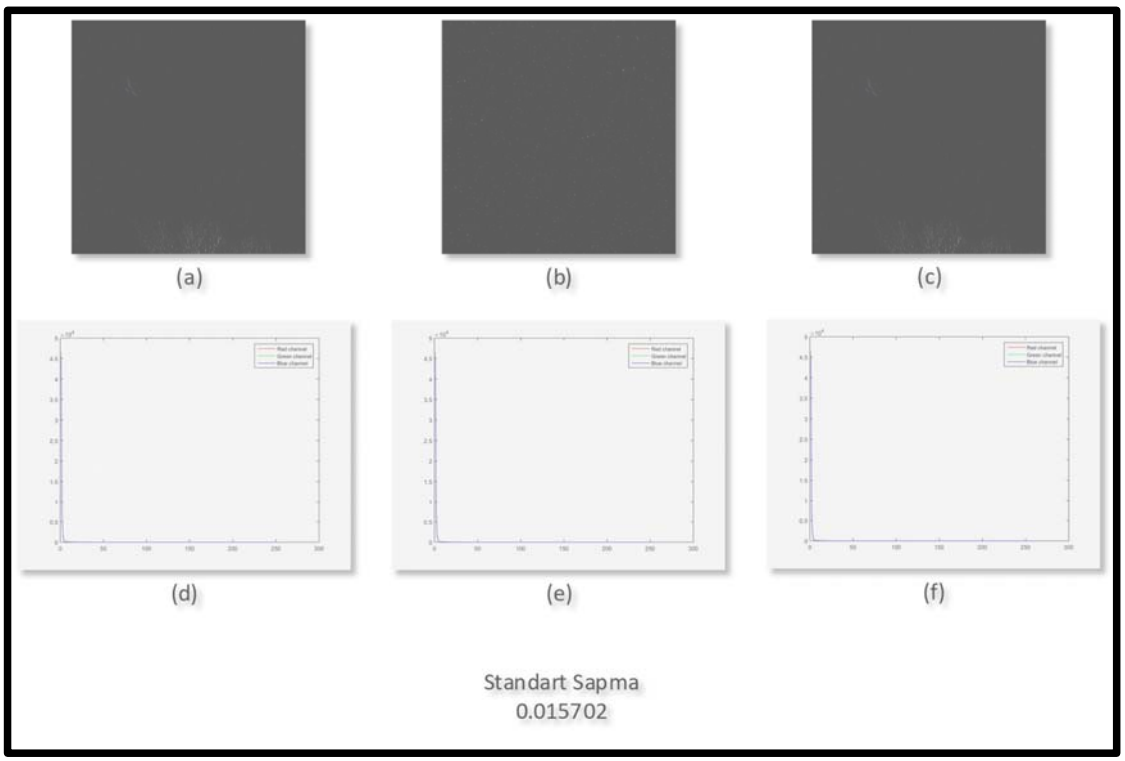
Şekil 3.13. Dikey Uçak Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)



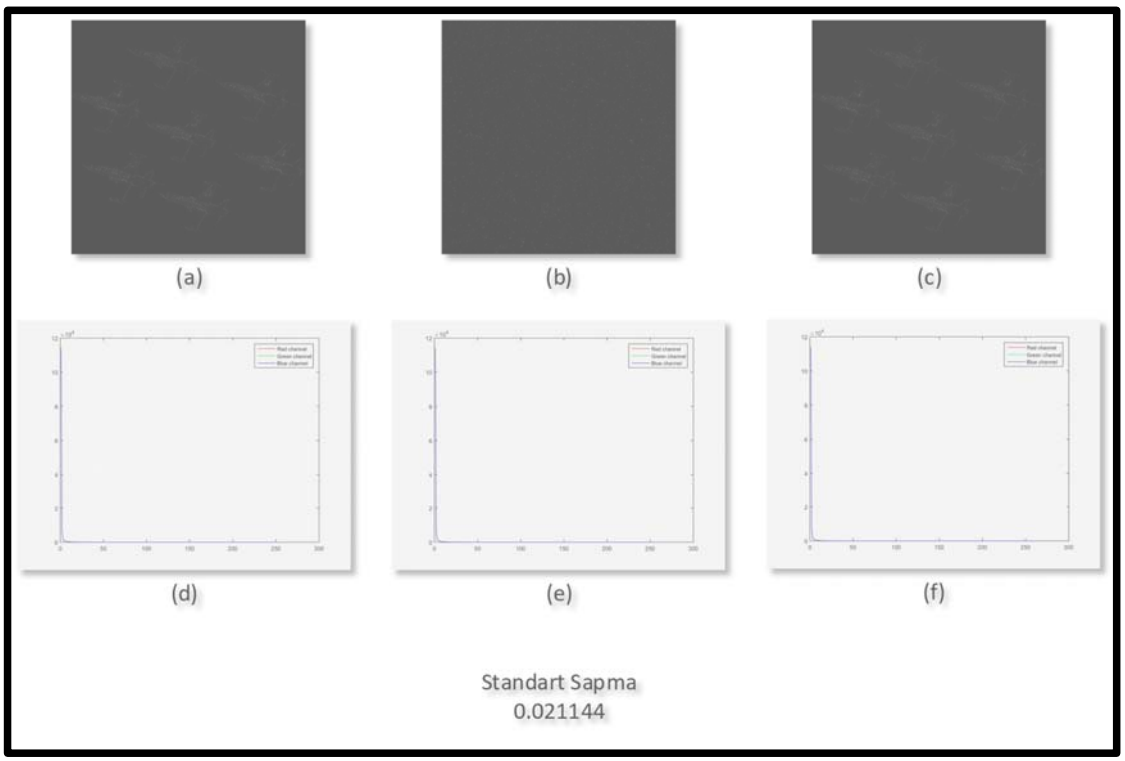
Şekil 3.14. Dikey Penguen Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)



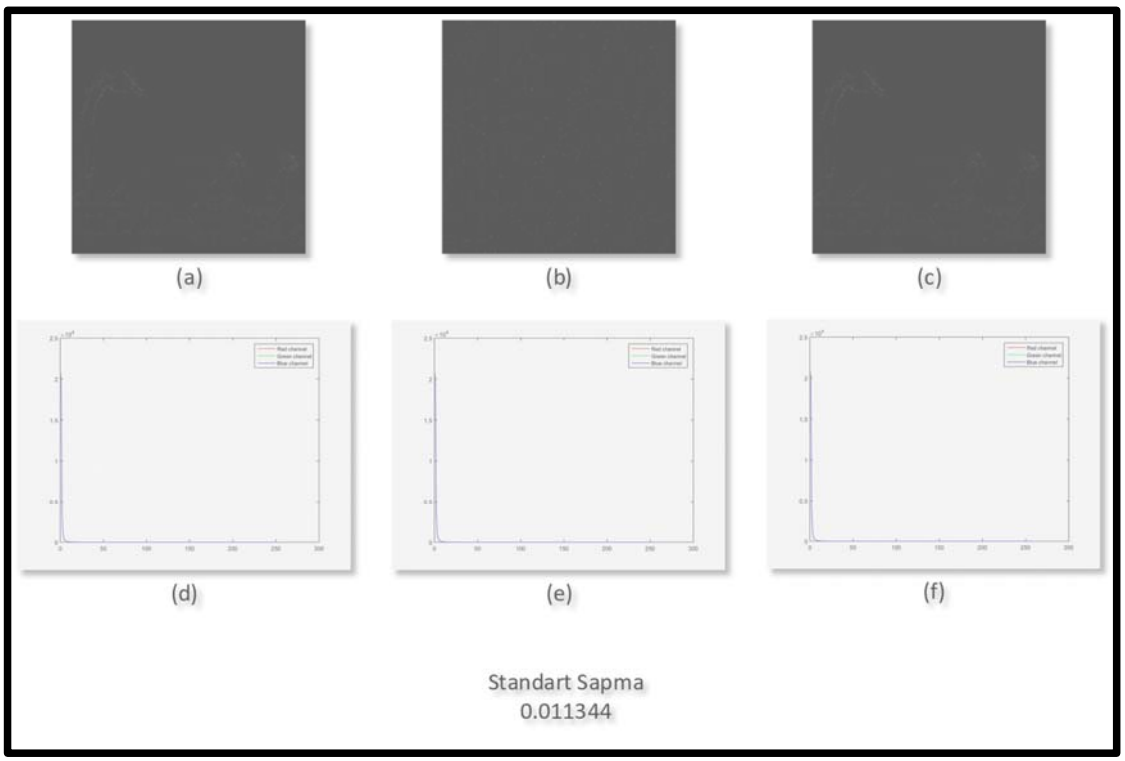
Şekil 3.15. Dikey İnsan Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)



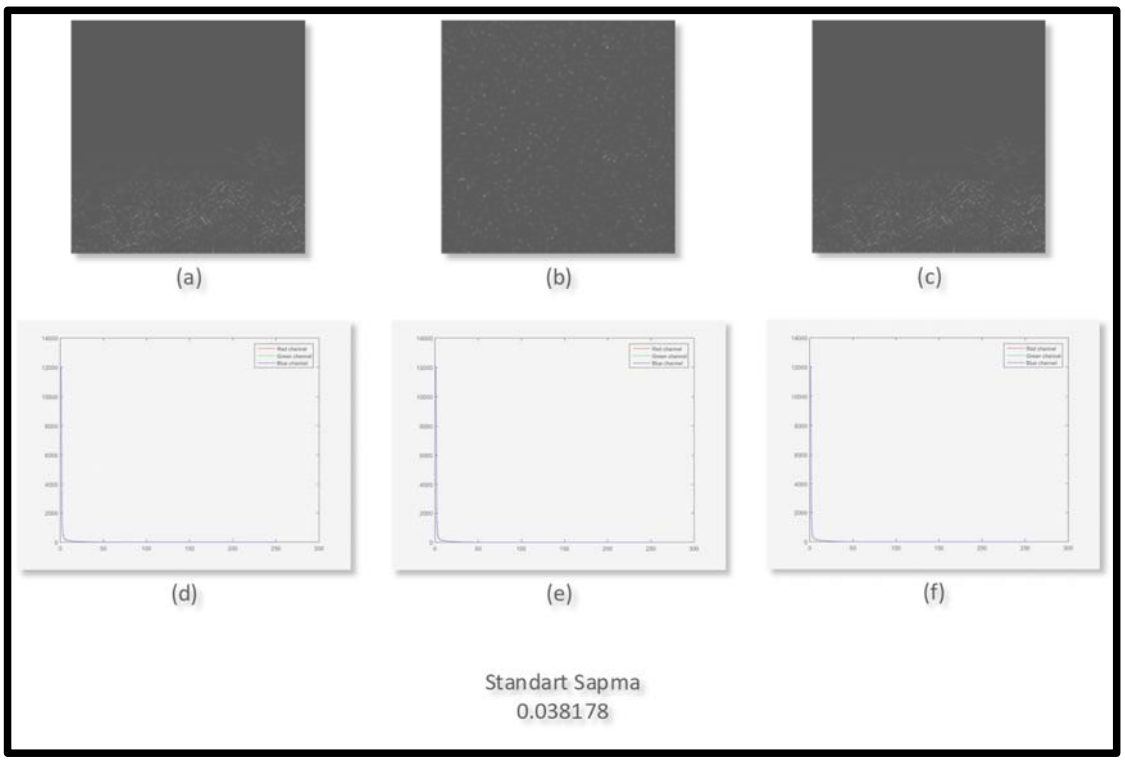
Şekil 3.16. Dikey Kuş Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)



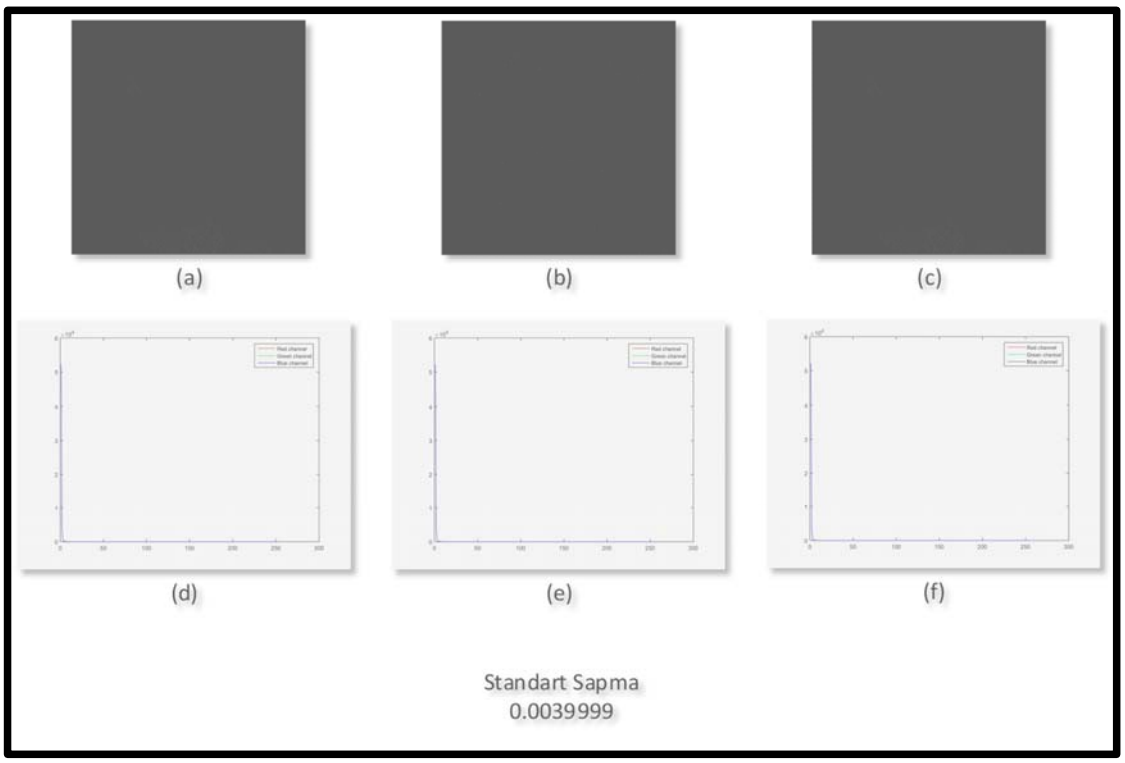
Şekil 3.17. Köşegen Uçak Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)



Şekil 3.18. Köşegen Penguen Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)



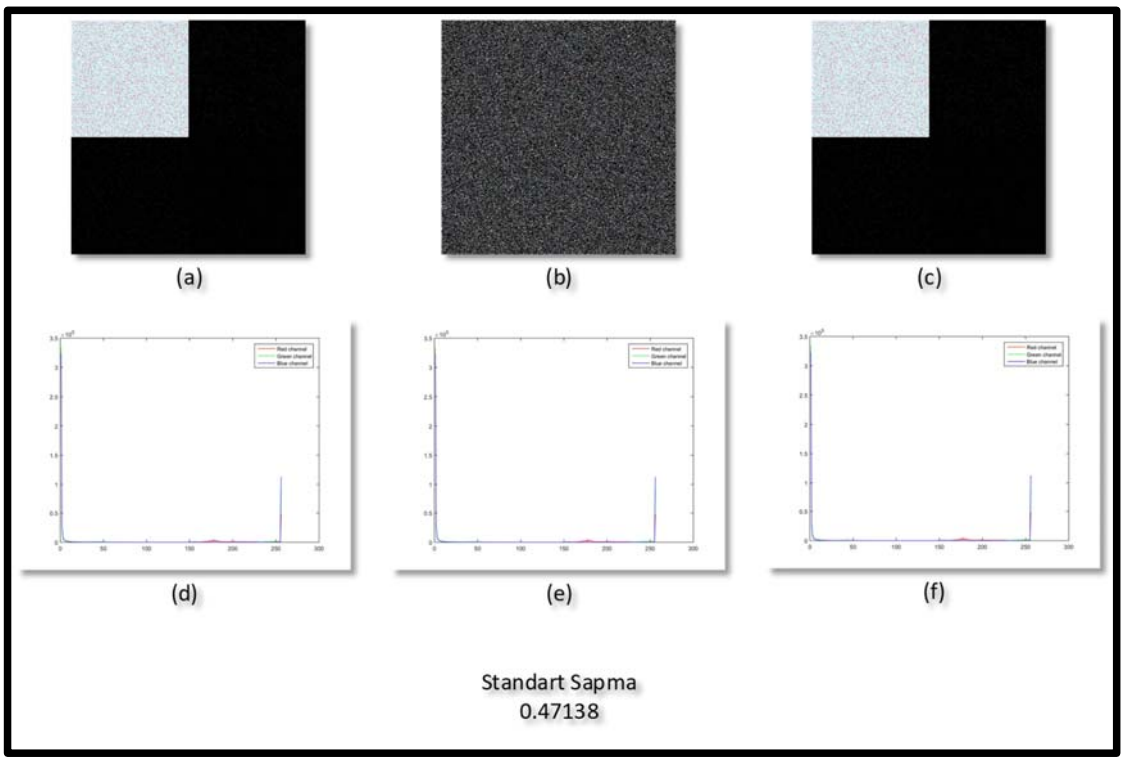
Şekil 3.19. Köşegen İnsan Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)



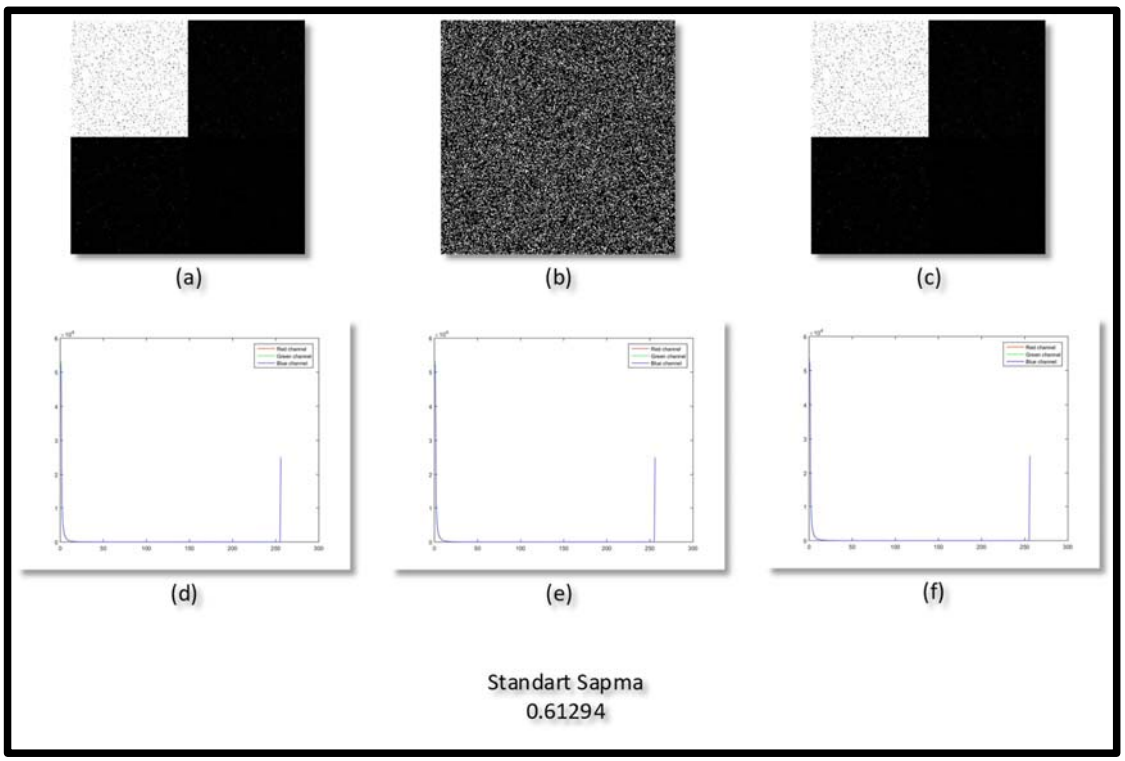
Şekil 3.20. Köşegen Kuş Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)

Bu üç bileşene bakıldığında (yaklaşık, yatay, dikey, köşegen) orijinal görseldeki renk değişimi keskin olan kısımlar görülebilmekte ve orijinal görsele göre siyah renk ağırlıklı bir veri kümesi elde edilmektedir. Bu nedenle orijinal renkli görsel ile bunlara ait histogramlar neredeyse tamamen farklıdır ve bu katsayılar ihmal edilebilir.

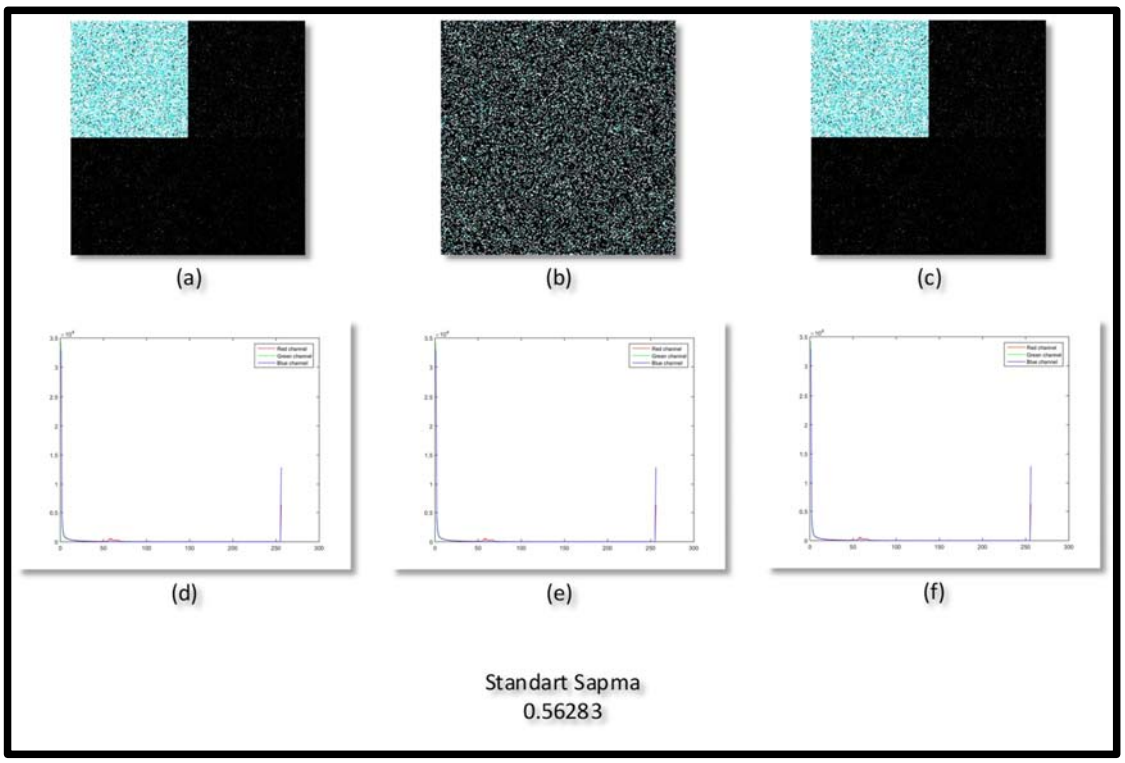
Uygulamanın devamında, yaklaşık (sol üst), yatay (sağ üst), dikey (sol alt) ve köşegen (sağ alt) detayları birleştirilerek dört farklı görüntü için elde edilen şifreli görüntülerin, MxN boyutundaki orijinal görsel ile aynı piksel sayısına sahip oldukları görülmektedir (Şekil 3.21, Şekil 3.22, Şekil 3.23, 3.24).



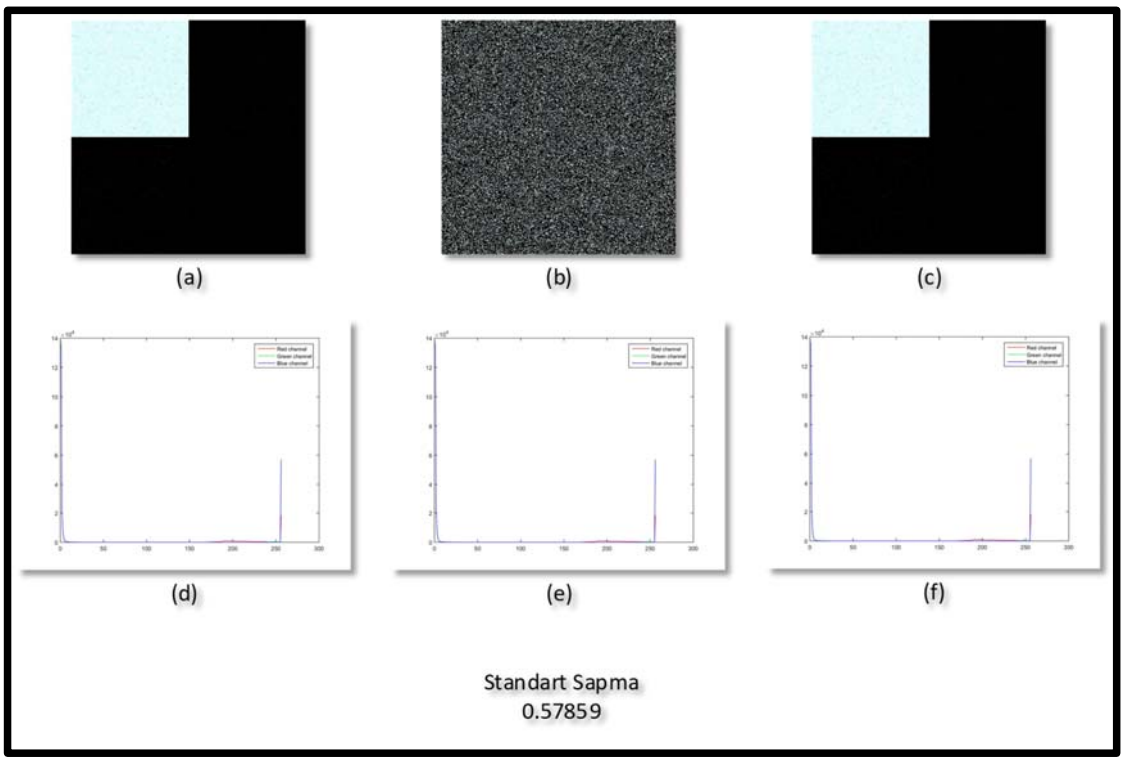
Şekil 3.21. AHVD Uçak Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)



Şekil 3.22. AHVD Penguen Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)



Şekil 3.23. AHVD İnsan Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)



Şekil 3.24. AHVD Kuş Görüntüsü Detayı ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)

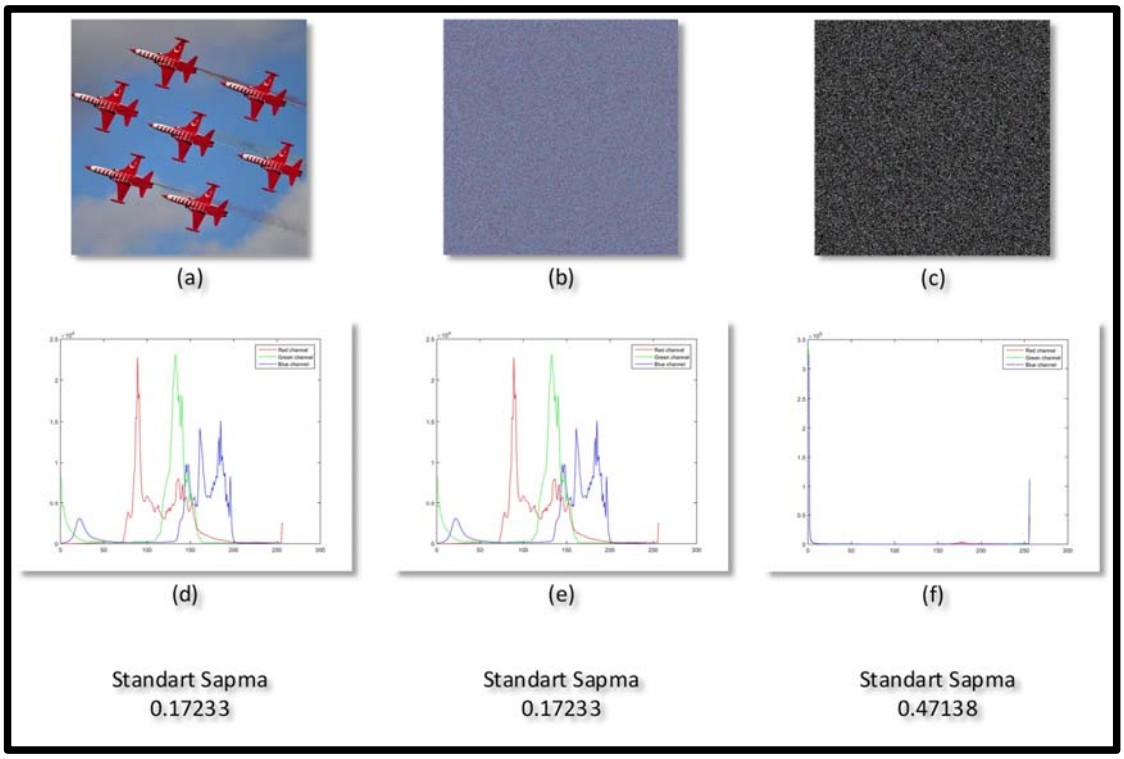
Orijinal görsellere ait şifrelenmiş ve birleştirilmiş dalgacık bileşen görselleri incelendiğinde (Şekil 3.21a, Şekil 3.22a, Şekil 3.23a, Şekil 3.24a), yaklaşık bileşen görselinde siyah ve beyaz renk yoğunlukları dışında bir veri kümesi bulunduğu için ayrık dalgacık dönüşümü uygulanmış olduğu tahmin edilebilmektedir. Bu nedenle daha anlaşılmaz bir görüntü elde edebilmek için şifrelenmiş ve birleştirilmiş dalgacık bileşenlerine de bütünüyle bir rastgele permütasyon işlemi uygulanarak Şekil 3.21, Şekil 3.22b, Şekil 3.23b, 3.24b'deki görseller elde edilmiştir.

Şekil 3.21b, Şekil 3.22b, Şekil 3.23b, 3.24b'deki görseller incelendiğinde, görsel üzerindeki renk dağılımları homojen olduğundan, orijinal görselin bir dönüşüm uygulanarak şifrelenip şifrelenmediği anlaşılmamaktadır. Bu nedenle de şifrenin çözülebilmesi için ters ayrık dalgacık dönüşümünün uygulanması gerekliği de tahmin edilememektedir. Göndericinin göndermiş olduğu şifreli görsel ile birlikte, katsayıların şifrelerini çözmek için gönderilen 4 ayrı şifre ve ayrık dalgacık dönüşümü katsayılarının birlikte oluşturduğu

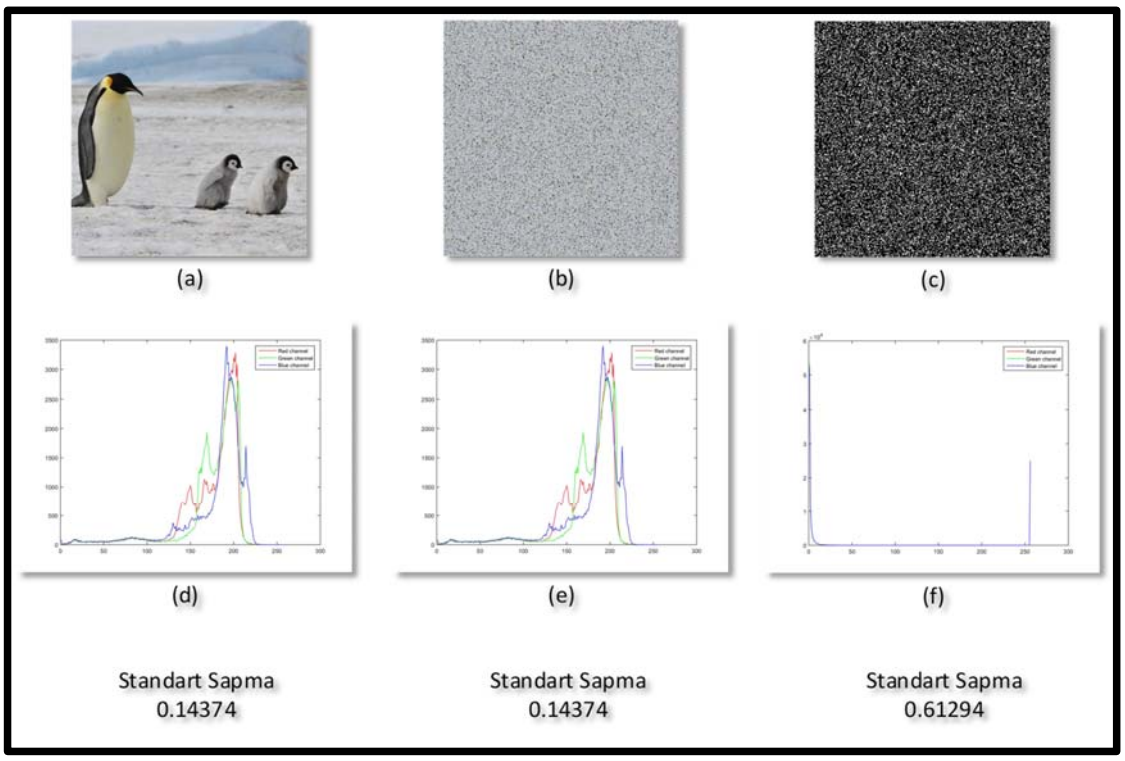
AHVD görselini çözebilmek için gönderilen 1 şifrenin, şifreli görselin anlamlı haline tekrar geri döndürülebilmesini oldukça zorlaştırmaktadır.

3.3. Basit ile Dalgacık Şifreleme Metotlarının Karşılaştırılması

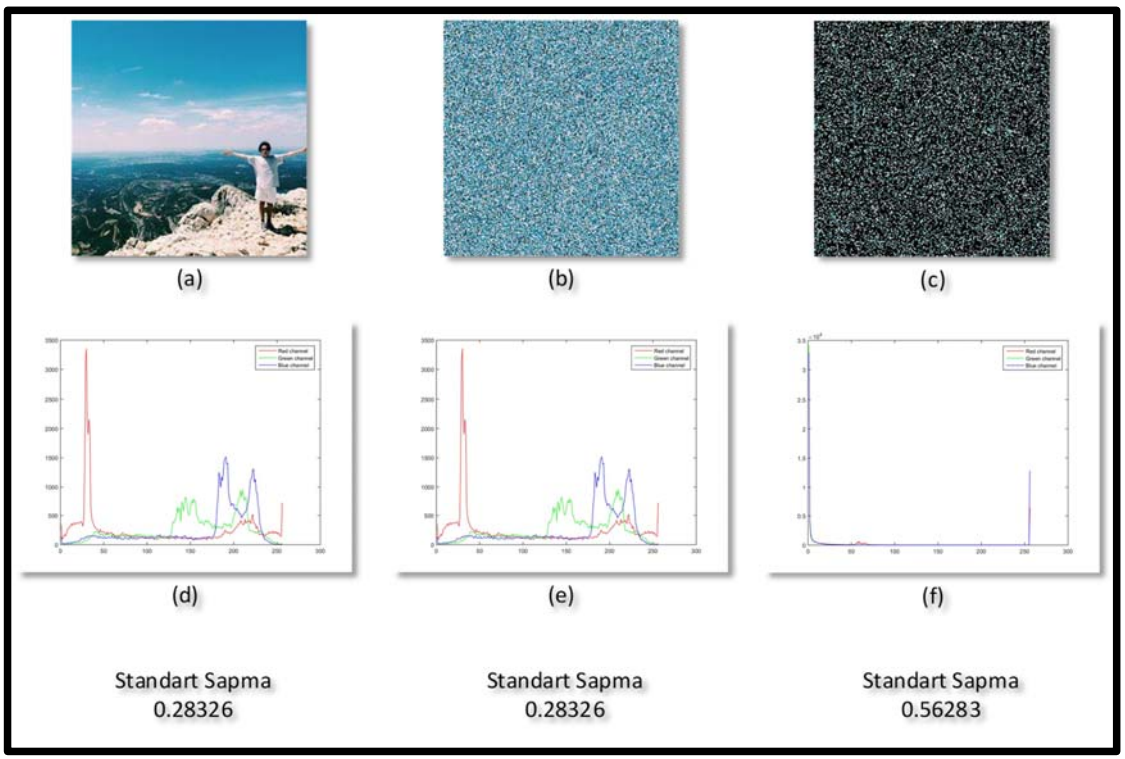
Orijinal görüntü ve rastgele permütasyon ile şifrelenmiş dört farklı görüntünün histogramları kıyaslandığında, iki görüntü arasında sadece piksellerin rastgele yer değiştirmesinden dolayı bir değişiklik görülmemektedir (Şekil 3.25a, Şekil 3.25b, Şekil 3.25d, Şekil 3.25e, Şekil 3.26a, Şekil 3.26b, Şekil 3.26d, Şekil 3.26e, Şekil 3.27a, Şekil 3.27b, Şekil 3.27d, Şekil 3.27e, Şekil 3.28a, Şekil 3.28b, Şekil 3.28d, Şekil 3.28e).



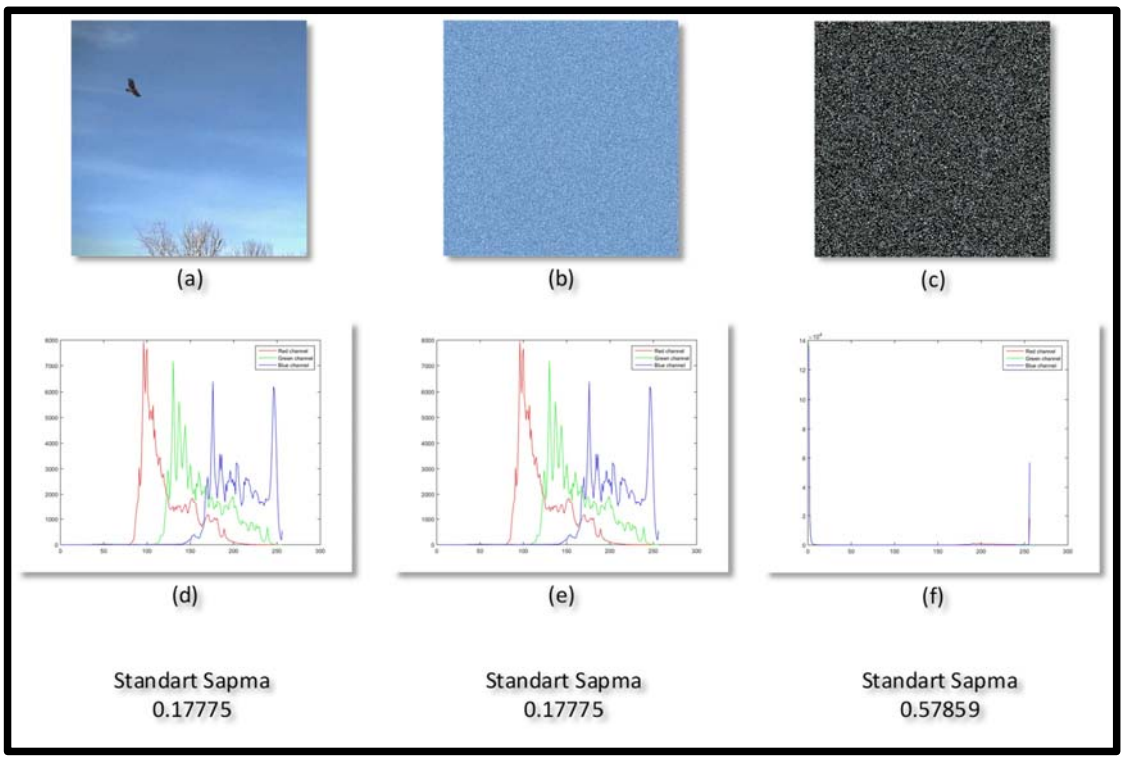
Şekil 3.25. Orijinal Uçak Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Uçak Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Uçak Görüntüsü ve Histogramı (c,f)



Şekil 3.26. Orijinal Penguen Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Penguen Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Penguen Görüntüsü ve Histogramı (c,f)



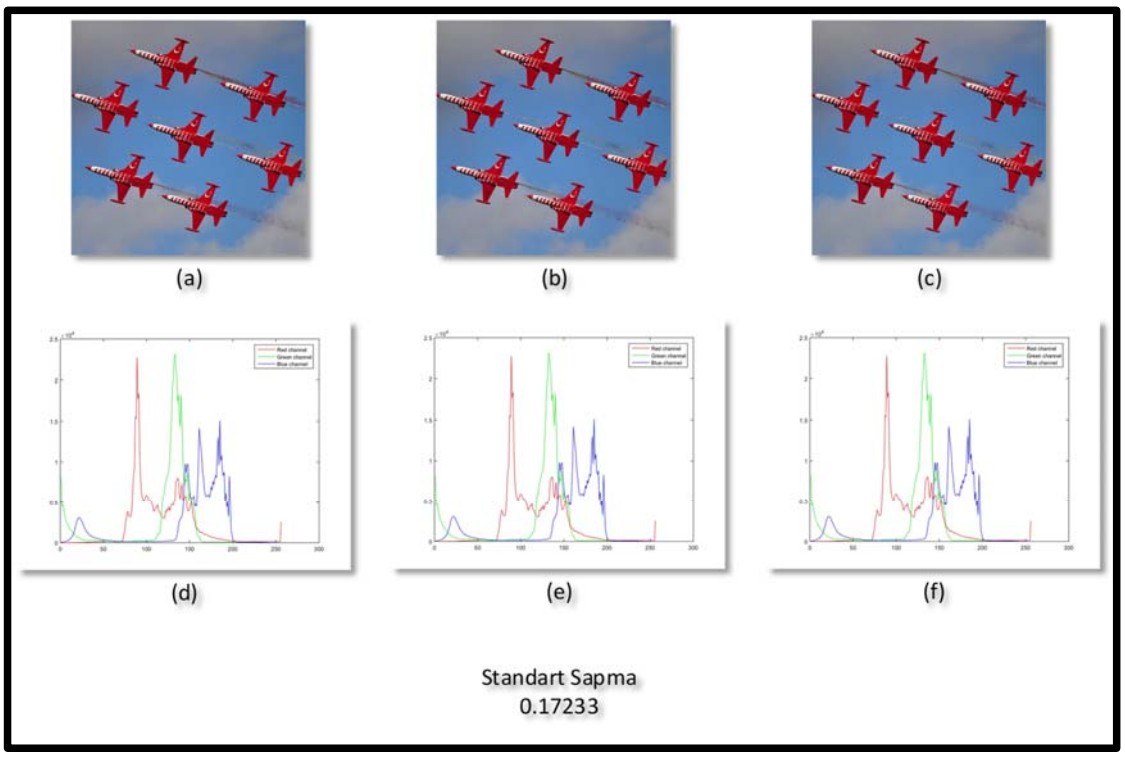
Şekil 3.27. Orijinal İnsan Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş İnsan Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelenmiş İnsan Görüntüsü ve Histogramı (c,f)



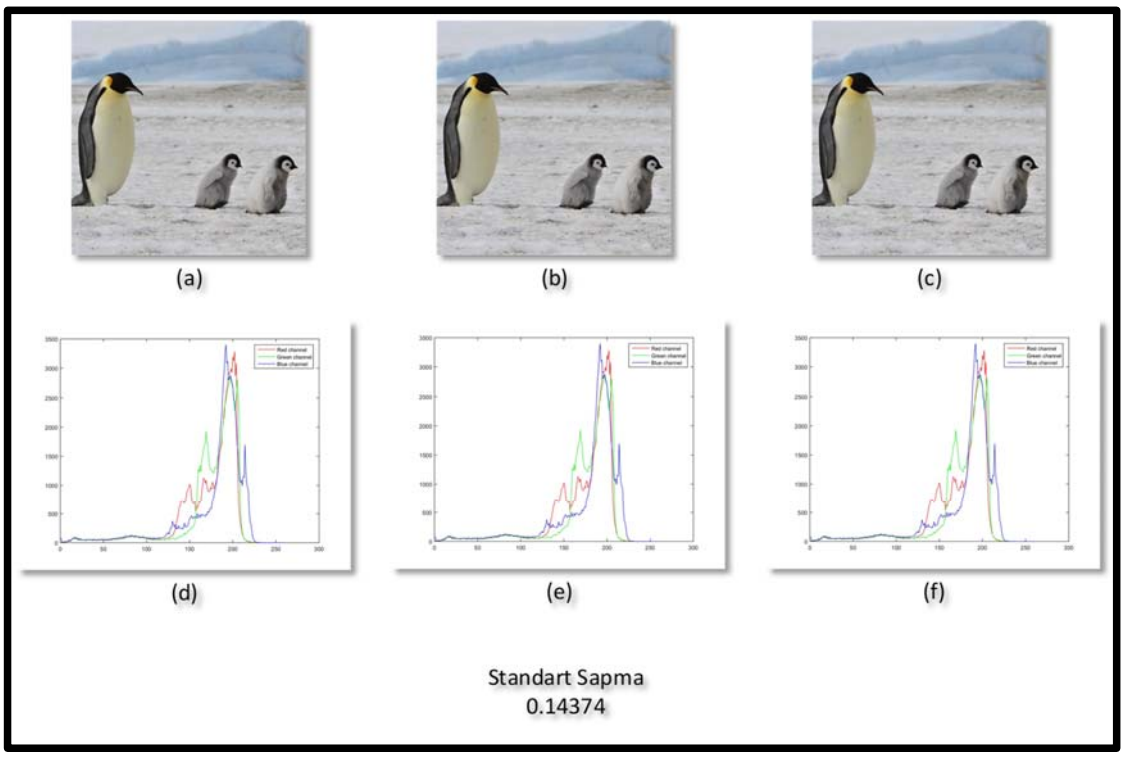
Şekil 3.28. Orijinal Kuş Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifrelenmiş Kuş Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Kuş Görüntüsü ve Histogramı (c,f)

Diğer taraftan; rastgele permütasyon ile şifrelenmiş görüntü ve ayrık dalgacık dönüşümü uygulanarak şifrelenmiş görüntü arasında ise oldukça büyük bir fark olduğu görülmekte (Şekil 3.25b, Şekil 3.25c, Şekil 3.25e, Şekil 3.25f, Şekil 3.26b, Şekil 3.26c, Şekil 3.26e, Şekil 3.26f, Şekil 3.27b, Şekil 3.27c, Şekil 3.27e, Şekil 3.27f, Şekil 3.28b, Şekil 3.28c, Şekil 3.28e, Şekil 3.28f), bu fark neticesinde de şifrelenmiş görüntüden, orijinal görüntüye ait renk dağılımı hakkında bir bilgi edinilememektedir.

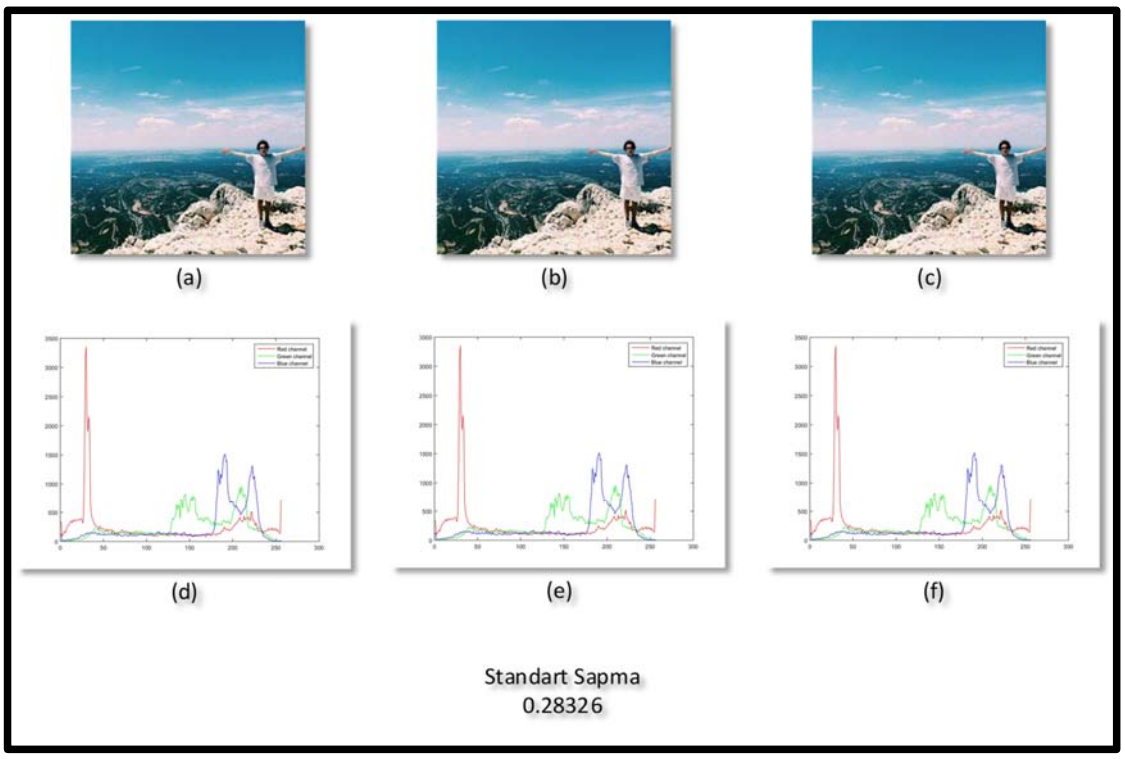
Çalışmanın devamında, şifrelenmiş görüntülerin tekrar orijinal görüntüye dönelebilmesi için aynı sıralama vektörü kullanılmış ve dört farklı görsel için bu görüntüler elde edilmiştir (Şekil 3.29b, Şekil 3.30b, Şekil 3.31b, Şekil 3.32b).



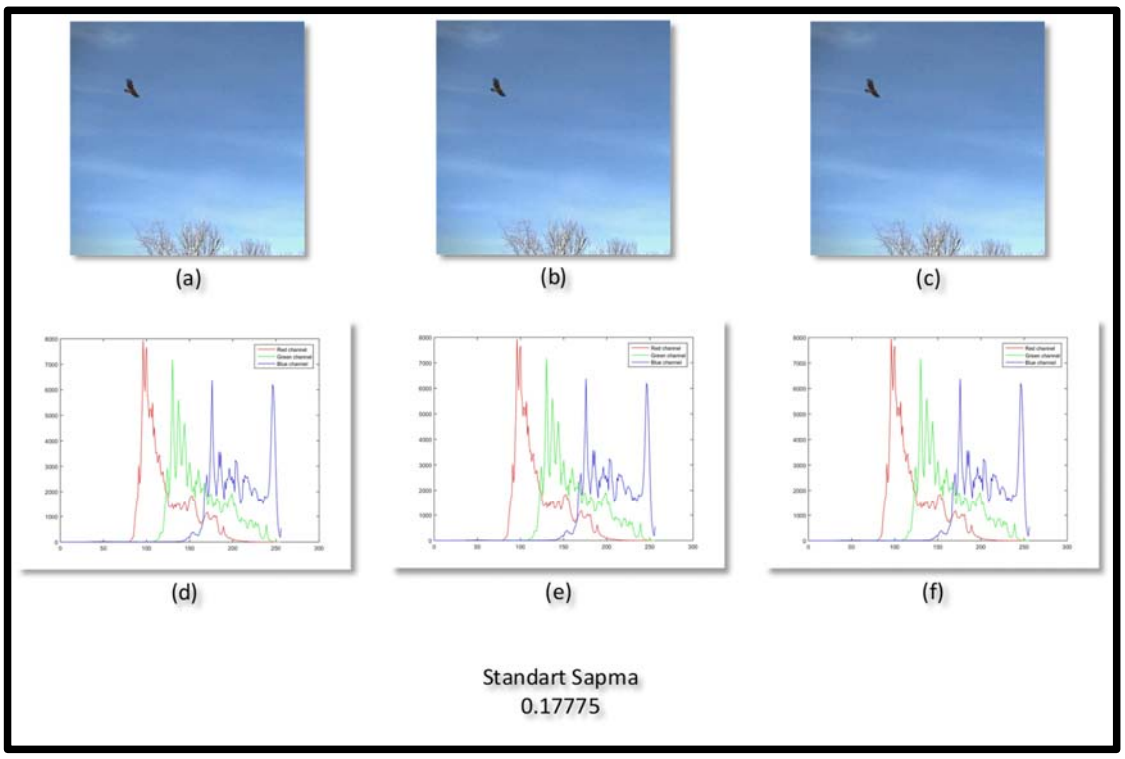
Şekil 3.29. Orijinal Uçak Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelendikten Sonra Şifresi Çözülmüş Uçak Görüntüsü ve Histogramı (c,f)



Şekil 3.30. Orijinal Penguen Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelendikten Sonra Şifresi Çözülmüş Penguen Görüntüsü ve Histogramı (c,f)



Şekil 3.31. Orijinal İnsan Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelendikten Sonra Şifresi Çözülmüş İnsan Görüntüsü ve Histogramı (c,f)



Şekil 3.32. Orijinal Kuş Görüntüsü ve Histogramı (a,d), Rastgele Permütasyon ile Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (b,e), Ayrık Dalgacık Dönüşümü İle Şifrelendikten Sonra Şifresi Çözülmüş Kuş Görüntüsü ve Histogramı (c,f)

Şekil 3.29, Şekil 3.30, Şekil 3.31, Şekil 3.32'de görüldüğü gibi, orijinal görüntüler ile şifreleri çözülmüş görüntüler arasında fark oluşmamıştır. Bu da şifrelenmiş görüntülerin tekrar eski haline dönüşümünün başarı ile sonuçlandığı anlamına gelmektedir.

Yine aynı şekilde, ayrık dalgacık dönüşümü uygulandıktan sonra şifrelenmiş dört farklı görüntüye ters ayrık dalgacık dönüşümü uygulanarak aynı sıralama vektörü kullanılmış ve böylece orijinal görüntülere tekrar ulaşılmıştır (Şekil 3.29c, Şekil 3.30c, Şekil 3.31c, Şekil 3.32c).

Basit şifreleme metodu ve dalgacık şifreleme metodu karşılaştırıldığında, iki farklı şifreleme yöntemi ile orijinal görseller arasındaki belirsizlik, formül (3.1)'deki denklem uygulanarak entropi değerleri elde edilmiştir.

$$H(m) = -\sum_{i=0}^{2N-1} P(m_i) log_2[P(m_i)] \quad (3.1)$$

$P(m_i)$: m_i' nin olasılığı

$H(m)$: entropi değeri

Bu sonuca göre, ayrık dalgacık dönüşümü ile şifrelenmiş görüntü ve orijinal görüntü entropileri arasında farklılık bulunmuştur (Şekil 3.33, Şekil 3.34, Şekil 3.35, Şekil 3.36).

Entropi ile görüntünün bilgi içeriği ölçülür. Bu ölçüm, bilgi kaynağının ortalama belirsizliği olarak söylenebilir. Görüntülerde uygulanan entropi, piksellerin uyum sağlayabileceklerine karşılık gelen yoğunluk seviyeleri olarak tanımlanır. Görüntü detayları entropi sayesinde daha iyi karşılaştırılabilir. 8-bitlik bir görüntü, piksel başına 256 değer alabileceğinden her bir değer için eşit olasılık olduğu zaman optimum değer 8'dir. Buna göre 8'e yakın bulunan her entropi değeri, original görüntüye yakınsamaktadır. Sonuçlar göstermiştir ki; ayrık dalgacık dönüşümü ile şifrelenmiş görüntülerden elde edilen entropi değerleri, her görsel için 8'den küçük ve 2.2546 ile 3.3883 arasında değişmektedir.

Görüntü Tipi		Entropi Değerleri
Orijinal Görüntü		7.1036
Rastgele Permutasyon İle Şifrelenmiş Görüntü		7.1036
Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü		2.5312

Şekil 3.33. Uçak Görüntüsü Entropi Değerleri Karşılaştırma Tablosu

Uçak temalı görsel için Şekil 3.33'e bakıldığında ayrık dalgacık dönüşümü ile şifrelenmiş görüntünün entropi değeri; rastgele permütasyon ile şifrelenmiş görüntünün entropi değerine göre az olması, görüntünün daha az bilgiye sahip olduğunu belirtmektedir.

Görüntü Tipi		Entropi Değerleri
Orijinal Görüntü		6.5759
Rastgele Permutasyon İle Şifrelenmiş Görüntü		6.5759
Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü		2.5635

Şekil 3.34. Penguen Görüntüsü Entropi Değerleri Karşılaştırma Tablosu

Şekil 3.34'de verilen sonuçlar incelendiğinde, sadece rastgele permütasyon kullanılarak zaman uzayında şifrelenmiş penguen temalı görüntü ile orijinal görüntü entropi değerinde bir farklılık görülmemektedir. Ayrık dalgacık dönüşümü ile şifrelenmiş görüntünün entropi değeri; rastgele permütasyon ile şifrelenmiş görüntünün entropi değerine göre küçük olarak bulunmuştur. Bu sonuca göre, ayrık dalgacık dönüşümü ile şifrelenmiş görüntünün, sadece rastgele permütasyon kullanılarak şifrelenmiş görüntüye kıyasla daha az bilgiye sahip olduğu sonucuna ulaşılmıştır.

Görüntü Tipi		Entropi Değerleri
Orijinal Görüntü		7.6712
Rastgele Permutasyon İle Şifrelenmiş Görüntü		7.6712
Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü		3.3883

Şekil 3.35. İnsan Görüntüsü Entropi Değerleri Karşılaştırma Tablosu

Şekil 3.35'de verilen, insan temalı orijinal görüntü ile rastgele permütasyon kullanılarak şifrelenmiş görüntü arasındaki karşılaştırma sonucu, insan görüntüsünde de, diğer görüntülerde olduğu gibi bir fark görülmemektedir. 8-bitlik görüntünün piksel başına 0 – 255 aralığındaki her bir değerinin eşit olasılıktaki optimum değeri 8 olabilmektedir. Şekil 3.35'de gösterilen 7.6712 değerleri ise 8 değerine yakınsamaktadır. Fakat ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntünün entropi değerinin 8' den oldukça uzak olan 3.3883 değerine ulaştığı görülmektedir.

Görüntü Tipi		Entropi Değerleri
Orijinal Görüntü		7.3172
Rastgele Permutasyon İle Şifrelenmiş Görüntü		7.3172
Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü		2.2546

Şekil 3.36. Kuş Görüntüsü Entropi Değerleri Karşılaştırma Tablosu

Diğer görsellerde de olduğu gibi kuş temalı orijinal görselin ve şifrelenmiş görüntülerinin entropi değerleri Şekil 3.36'da gösterilmiştir. Ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntü, maksimum entropi değeri olan 8'den daha düşük bir değere sahip olduğu görülmektedir (2.2546). Sadece rastgele permütasyon kullanılarak şifrelenmiş görüntüye kıyasla, ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntünün daha az bilgiye sahip ve aralarında oldukça az benzerlik olduğu incelenmiştir.

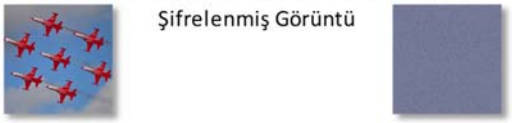
Orijinal görüntü ve şifreli görüntü arasındaki farklı oran değerleri MSE (mean square error) hesabı kullanılarak kümülatif kare hatası elde edilir. MSE değeri ne kadar düşük olursa hata oranı da o kadar düşük olur.

$$MSE = \frac{1}{MxN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[f(i,j) - \hat{f}(i,j)\right]^2 \qquad (3.2)$$

$f(i,j)$: Orijinal görüntü

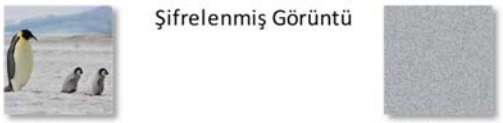
$\hat{f}(i,j)$: Şifrelenmiş görüntü

Dört farklı görsele uygulanmış iki farklı şifreleme yöntemi için ortalama kare hataları hesaplanmış ve sonuçlar Şekil 3.37, Şekil 3.38, Şekil 3.39, Şekil 3.40' da gösterilmiştir.

Ortalama Kare Hatası (MSE)				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	0.2393	0.23823	0.27618	0.42271
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	0.026822	0.031811	0.053427	0.070242

Şekil 3.37. Uçak Görüntüsü Ortalama Kare Hatası

Uçak temalı görselde uygulanan iki farklı metodun sonucunda elde edilen şifreli görüntüler ile orijinal görüntü arasındaki ortalama kare hatasına bakıldığında, ayrık dalgacık yöntemi kullanılarak şifrelenmiş görüntü ile orijinal görüntü arasındaki hata oranı (0.2393), basit şifreleme metoduna göre (0.026822) yaklaşık 10 kat yüksek bulunmuştur (Şekil 3.37).

Ortalama Kare Hatası (MSE)				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	0.39135	0.49686	0.51317	0.53481
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	0.038282	0.03971	0.038051	0.045936

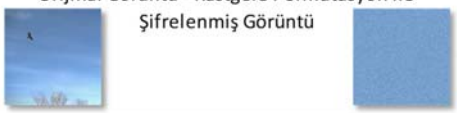
Şekil 3.38. Penguen Görüntüsü Ortalama Kare Hatası

Penguen temalı orijinal görüntü ile iki farklı biçimde şifrelenmiş görüntülerin aralarındaki kümülatif kare hatasına bakıldığında, ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntünün, orijinal görsele kıyasla, aralarındaki hata oranının yaklaşık olarak 10 kat olduğu Şekil 3.38'de görülmektedir.

Ortalama Kare Hatası (MSE)				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	0.31204	0.38683	0.45374	0.5514
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	0.10995	0.2161	0.090404	0.10255

Şekil 3.39. İnsan Görüntüsü Ortalama Kare Hatası

İnsan temalı orijinal görüntünün, sadece rastgele permütasyon kullanılarak şifrelenmiş görüntü ile arasındaki ortalama kare hatasının (0.10995); orijinal görüntü ile ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntü arasındaki ortalama kare hatasından (0.31204) daha düşük olduğu Şekil 3.39'a bakıldığında görülmektedir. MSE oranının düşüklüğü neticesinde hata oranının da düşük olduğu, bunun sonucunda orijinal görüntüyle rastgele permütasyon kullanılarak şifrelenmiş görüntünün birbirlerine benzerlik oranlarının yüksek olduğu görülebilmektedir.

Ortalama Kare Hatası (MSE)				
Görüntü Tipi	Gri Ölçek	R	G	B
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	0.32011	0.24588	0.42053	0.67863
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	0.025337	0.023139	0.028027	0.02468

Şekil 3.40. Kuş Görüntüsü Ortalama Kare Hatası

Şekil 3.40'da sunulan kuş temalı görüntüde de benzer karşılaştırma yapılmış, şifreleme işlemi yapılan diğer 3 görselde de olduğu gibi, orijinal görüntü ile rastgele permütasyon uygulanarak şifrelenmiş görüntü arasındaki hata oranı (0.025337), ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntüye oranla (0.32011) daha düşüktür. MSE oranının ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntüde yüksek olması, şifrelenmiş görüntünün orijinal görüntüye kıyasla daha az benzerlik gösterdiğini kanıtlamaktadır.

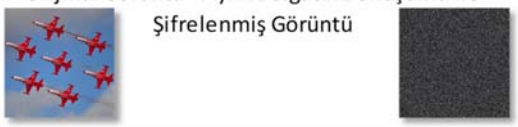
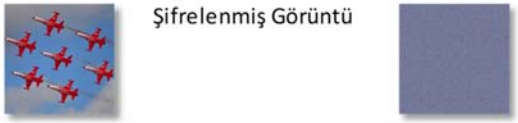
PSNR (peak signal noise ratio) kullanılarak orijinal görüntü ile şifreli görüntü arasındaki düzensizlik oranı elde edilir. Bu oran, renkli görüntülerde R, G ve B renk kanallarının ağırlıklı ortalaması alınarak hesaplanır. Genellikle insan gözü G kanalını daha iyi algıladığı için yeşil kanal doğru sonuca daha yakın çıkar.

Orijinal görüntü ile sıkıştırılmış veya tekrar yapılandırılmış görsel arasındaki kaliteyi ölçmek için PSNR ile tepe noktasına göre hata ölçümü denklem (3.3) kullanılarak hesaplanır. Yüksek PSNR değeri daha kaliteli bir görsel olduğunu gösterir [14, 19]. Bu değer desibel cinsinden ifade edilir.

$$PSNR = 10 \log_{10} \left[\frac{{max_f}^2}{MSE} \right] \tag{3.3}$$

M ile N değerleri (3.2) görüntü boyutunu ve max_f ise f görüntüsünün maksimum olasılık değerini ifade eder.

Dört farklı görüntünün MSE değerlerine istinaden hesaplanan PSNR oranları, Şekil 3.41, Şekil 3.42, Şekil 3.43, Şekil 3.44'de gösterilmiştir.

Tepe Sinyal Gürültü Oranı (PSNR)				
Görüntü Tipi	Gri Ölçek	R	G	B
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	54.3753	54.3948	53.7529	51.9044
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	63.8799	63.139	60.8872	59.6988

Şekil 3.41. Uçak Görüntüsü Tepe Sinyal Gürültü Oranı

Şekil 3.41'de, uçak temalı original görsel ile iki farklı metot kullanılarak elde edilmiş şifreli görselleri arasındaki PSNR değerleri görülmektedir. Ayrık dalgacık analizi kullanılarak şifrelenmiş görüntü (54.3753), sadece rastgele permütasyon kullanılarak şifrelenmiş görüntüye (63.8799) göre daha düşük bir PSNR değerine sahiptir. Bu nedenle şifrelenmiş görüntünün original görselden daha düşük bir kalitede olduğu anlaşılabilmektedir.

Tepe Sinyal Gürültü Oranı (PSNR)				
Görüntü Tipi	Gri Ölçek	R	G	B
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	52.2392	51.2025	51.0622	50.8828
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	62.3349	62.1758	62.3611	61.5433

Şekil 3.42. Penguen Görüntüsü Tepe Sinyal Gürültü Oranı

Penguen görüntüsü için MSE değerine bağlı olarak hesaplanan PSNR değerleri Şekil 3.42'de gösterilmiştir. Orijinal görüntü ile iki farklı biçimde şifreleme uygulanarak yeniden yapılandırılmış görüntüler arasındaki düzensizlik oranı, PSNR, hesaplanmıştır. Bu oranlara bakıldığında ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntü ile orijinal görüntü arasındaki PSNR değeri (52.2392); sadece rastgele permütasyon işlemi uygulanarak şifrelenmiş görüntü ile orijinal görüntü arasındaki PSNR değerinden (62.3349) düşük olarak elde edilmiştir. Ayrık dalgacık dönüşümü ile şifrelenmiş görüntü kalitesi, sadece rastgele

permütasyon kullanılarak şifrelenmiş görüntünün kalitesinden daha düşük bir değere sahip olduğu görülmektedir.

Tepe Sinyal Gürültü Oranı (PSNR)				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü  	53.2227	52.2895	51.5967	50.7501
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü  	57.7529	54.8182	58.6029	58.0554

Şekil 3.43. İnsan Görüntüsü Tepe Sinyal Gürültü Oranı

İnsan temalı orijinal görüntü ile ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntü arasındaki PSNR değeri (53.2227); orijinal görüntü ile rastgele permütasyon kullanılarak şifrelenmiş görüntü arasındaki PSNR değerinden (57.7529) daha alçak olduğu Şekil 3.43'de görülmektedir. Düzensizlik oranının ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntüde daha düşük elde edilmesi, görüntü kalitesinin daha düşük olduğunu göstermektedir.

Tepe Sinyal Gürültü Oranı (PSNR)				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü  	53.1118	54.2575	51.9268	49.8484
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü  	64.1273	64.5214	63.689	64.2413

Şekil 3.44. Kuş Görüntüsü Tepe Sinyal Gürültü Oranı

Şekil 3.44'de kuş temalı orijinal görüntü ile iki farklı şifreleme yöntemi arasındaki PSNR değerleri gösterilmiştir. Ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntünün PSNR oranı (53.1118), sadece rastgele permütasyon kullanılarak şifrelenmiş görüntüye kıyasla (64.1273) daha düşüktür. Ayrık dalgacık dönüşümü ile şifrelenmiş görüntünün PSNR oranının diğer şifreleme yöntemine göre düşük olması, daha düşük görüntü kalitesinin elde edildiğini kanıtlamaktadır.

İki görsel arasındaki korelasyon katsayısı r, (-1,1) aralığında değer almaktadır. Bu değerin sıfıra yaklaşması, iki görüntü arasındaki farklılığın büyüklüğü ile doğru orantılıdır [7, 15].

Orijinal ve şifreli görüntü arasındaki her bir renk kanalı için çapraz korelasyon katsayıları formül (3.4)'e göre hesaplanabilir [17].

$$\Upsilon(x,y) = \frac{\sum_s \sum_t \left[f(s,t) - \bar{f}(s,t)\right][w(x+s,\ y+t) - \overline{w}]}{\left\{\sum_s \sum_t \left[f(s,t) - \bar{f}(s,t)\right]^2 \sum_s \sum_t [w(x+s,\ y+t) - \overline{w}]^2\right\}^{\frac{1}{2}}} \qquad (3.4)$$

x : Görüntünün eni; 0,1,2…M-1

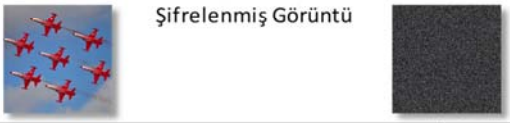
y : Görüntünün boyu; 0,1,2…N-1

$\bar{f}\ ve\ \overline{w}$: Orijinal ve şifreli görüntünün ortalaması

$f(s,t)$: Orijinal görüntünün (s,t) pozisyonundaki piksel değeri

$w(s,t)$: Şifreli görüntünün (s,t) pozisyonundaki piksel değeri

Görseller üzerinde uygulanan iki farklı şifreleme metodu arasındaki korelasyon hesaplamaları, her bir görsel için Şekil 3.45, Şekil 3.46, Şekil 3.47, Şekil 3.48'de gösterilmiştir.

Korelasyon				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Rastgele Permutasyon İle Şifrelenmiş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	0.00083551	-0.00164345	0.0010717	0.0003636
Rastgele Permutasyon İle Şifresi Çözülmüş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifresi Çözülmüş Görüntü 	0.99	0.99	0.99	0.99
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	-0.000025869	0.0014151	0.000049452	-0.00072902
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	0.001028	0.00098066	0.00021111	0.0018925

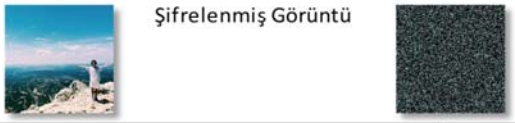
Şekil 3.45. Uçak Görüntüsü Korelasyon Karşılaştırma Tablosu

Şekil 3.45'de sunulan uçak temalı görsel için, rastgele permütasyon uygulanarak şifrelenmiş görüntü ile ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntü arasında korelasyon değerinde sıfıra yakınsama görülmektedir (0.00083551). İki farklı şifreleme yönteminin, original görüntü arasındaki çapraz korelasyon katsayılarına bakıldığında ise, ayrık dalgacık dönüşümü ile şifrelenmiş görüntünün korelasyon katsayısı (-0.000025869), rastgele permütasyon kullanılarak şifrelenmiş görüntünün korelasyon katsayına göre (0.001028) sıfıra daha da yakınsamaktadır. İki şifreli görüntünün de tekrar orijinal görüntüye geri çevrilmesi sonucunda ise aralarında bir fark bulunmadığı, korelasyon değerinin 1 olmasıyla ispatlanmıştır.

Korelasyon				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Rastgele Permutasyon İle Şifrelenmiş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	-0.0028373	-0.0036352	-0.0028572	-0.0011897
Rastgele Permutasyon İle Şifresi Çözülmüş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifresi Çözülmüş Görüntü 	0.99	0.99	0.99	0.99
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	-0.00175	-0.0011871	-0.0021231	-0.002854
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	-0.0019352	-0.0011009	-0.0020986	-0.0032278

Şekil 3.46. Penguen Görüntüsü Korelasyon Karşılaştırma Tablosu

İki farklı şifreleme yöntemi ile şifrelenmiş penguen temalı görüntüler arasında Şekil 3.46'da sıfıra yakınsama (-0.0028373) görülmektedir. Ayrık dalgacık dönüşümü ile şifrelenmiş görüntünün original görüntüyle çapraz korelasyon katsayısı (-0.00175) ve rastgele permütasyon kullanılarak şifrelenmiş görüntünün original görüntüyle çapraz korelasyon katsayına (-0.0019352) göre nispeten sıfıra daha çok yakınsamaktadır. Her iki şifrelenmiş görselin aralarındaki çapraz korelasyon katsayısı da sıfıra yakınsamaktadır. İki şifreli görüntünün de tekrar orijinal görüntüye geri çevrilmesi sonucunda ise aralarında bir fark bulunmadığı, korelasyon değerinin 1 olmasıyla ispatlanmıştır.

Korelasyon				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Rastgele Permutasyon İle Şifrelenmiş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	0.0001399	-0.0023724	-0.0024797	-0.0045033
Rastgele Permutasyon İle Şifresi Çözülmüş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifresi Çözülmüş Görüntü 	0.99	0.99	0.99	0.99
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü 	-0.0017147	0.0014409	-0.00012247	-0.00096616
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü 	0.0050199	0.0013212	0.0058884	0.0072555

Şekil 3.47. İnsan Görüntüsü Korelasyon Karşılaştırma Tablosu

Şekil 3.47'de orijinal görüntü ile ayrık dalgacık dönüşümü uygulanarak şifrelenmiş görüntü arasındaki sıfıra yakınsayan değer (-0.0017147); orijinal görüntü ile sadece rastgele permütasyon uygulanarak şifrelenmiş görüntü arasındaki sıfıra yakınsayan korelasyon değerinden (0.0050199) daha düşük olduğu görülmektedir.

Her iki şifreleme yönteminde de korelasyon değeri sıfıra yakınsamış olmasına rağmen, orijinal görüntü ile ayrık dalgacık dönüşümü uygulanarak şifrelenmiş görüntü arasındaki korelasyon değeri (-0.0017147) sıfıra daha çok yakınsamaktadır. Bu durum neticesinde ayrık dalgacık dönüşümü kullanılarak şifrelenmiş görüntü ile rastgele permütasyon kullanılarak şifrelenmiş görüntü arasındaki korelasyon da incelenerek (0.0001399) sıfıra yakınsama elde edildiği de yine Şekil 3.47'de gözlemlenmektedir.

Bu iki görüntü arasındaki korelasyon değeri sıfıra yakınsadığında görüntüler arasındaki farklılığın arttığı bilinmektedir. İki şifreli görüntünün de tekrar orijinal görüntüye geri çevrilmesi sonucunda ise aralarında bir fark bulunmadığı, korelasyon değerinin 1 olmasıyla ispatlanmıştır.

Korelasyon				
Görüntü Tipi	**Gri Ölçek**	**R**	**G**	**B**
Rastgele Permutasyon İle Şifrelenmiş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü  	0.00096984	-0.0012461	-0.000897	-0.00001075
Rastgele Permutasyon İle Şifresi Çözülmüş Görüntü - Ayrık Dalgacık Dönüşümü İle Şifresi Çözülmüş Görüntü  	0.99	0.99	0.99	0.99
Orijinal Görüntü - Ayrık Dalgacık Dönüşümü İle Şifrelenmiş Görüntü  	0.000432	0.00056935	0.00053753	0.0012789
Orijinal Görüntü - Rastgele Permutasyon İle Şifrelenmiş Görüntü  	0.0000096893	-0.00010534	0.00012294	0.00024259

Şekil 3.48. Kuş Görüntüsü Korelasyon Karşılaştırma Tablosu

Şekil 3.48'de her iki şifreleme yönteminin orijinal görseller ile arasında uygulanan çapraz korelasyon katsayıları (0.00096984) gösterilmiştir. Her iki görselde de korelasyon katsayıları sıfıra yakınsamaktadır. İki farklı şekilde şifrelenmiş görüntü arasındaki çapraz korelasyon katsayısı da sıfıra yakınsamaktadır.

2.60Ghz işlemci hızı olan ve 16 GB RAM bellekli bir bilgisayarda Mathworks firmasının (Mathworks, MA, USA) MATLAB R2015a programı kullanılarak tüm görsellerde şifreleme ve şifre çözme işlemlerinde harcanan zamanlar Şekil 3.49'da gösterilmiştir. Bu değerlere bakıldığında görüntünün boyutuyla doğru orantılı olarak işlem süresi de yükselmektedir. Rastgele permütasyon ile şifreleme işlemi dalgacık dönüşümü ile şifreleme işleminden uçak, penguen, insan ve kuş görselleri için sırasıyla %30, %10, %9, %7 daha hızlı işlem gerçekleştirdiği görülmektedir. Şifre çözülürken geçen süre ise yaklaşık olarak aynıdır (Şekil 3.49).

Harcanan Zaman					
Görüntü Tipi	Rastgele Permütaston Şifreleme (s)	Rastgele Permütaston Şifre Çözme (s)	Dalgacık Dönüşümü (s)	Dalgacık Dönüşümü Şifreleme (s)	Dalgacık Dönüşümü Şifre Çözme (s)
Uçak (720x720)	0.1117480	0.1169580	0.2499680	0.1190260	0.1246700
Penguen (330x330)	0.0151970	0.0247040	0.1216890	0.0282230	0.0220090
İnsan (256x256)	0.0121730	0.0179940	0.1155400	0.0181070	0.0161120
Kuş (478x478)	0.0379060	0.0561830	0.1538160	0.0662010	0.0589190

Şekil 3.49. İşlem Süreleri

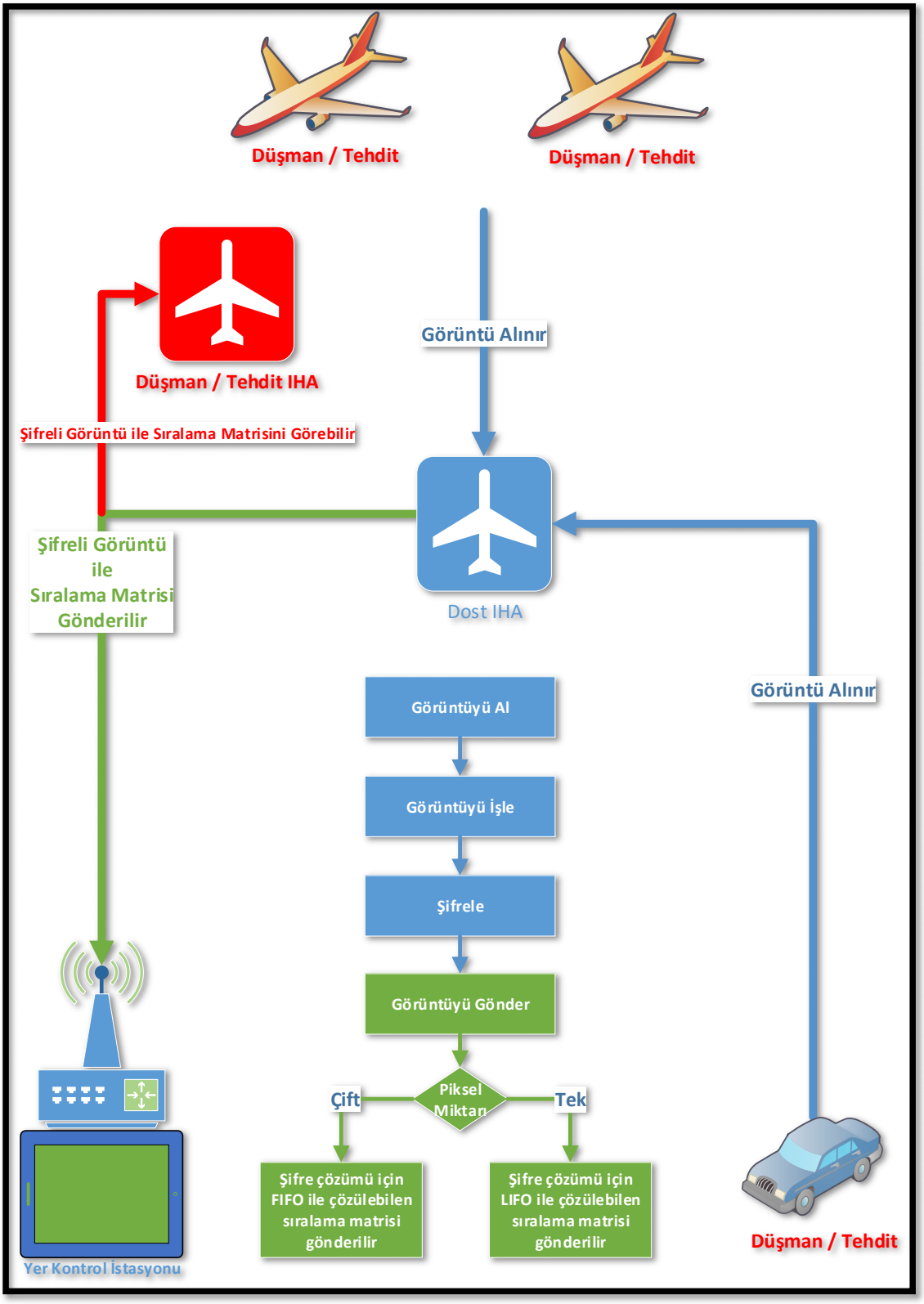
3.4. Örnek Bir Uygulama

Askeri alanda uygulanabilecek bir senaryoda teorik olarak şifreleme işlemi gerçekleştirilmiştir. Bu şifreleme işlemindeki amaç; dost IHA (insansız hava aracı) vasıtasıyla gönderilecek görüntü verisinin şifrelenerek güvenli bir biçimde yer kontrol istasyonuna iletilmesidir.

Düşman veya tehdit hava ve kara unsurlarının görüntü verileri, dost IHA tarafından elde edilir. Elde edilen bu gizli veri, dost IHA tarafından işleme sokulur. Görüntüler önce dalgacık dönüşümü ile detaylarına ayrılır. Daha sonra bu detaylara rastgele permütasyon işlemi uygulanır. Şifrelenmiş detaylar birleştirilerek tekrar rastgele permütasyon işlemi uygulanır ve 5 farklı şifre çözümü için kullanılacak vektör anahtarı elde edilir.

Dost IHA, elde ettiği şifreli görüntü ile birlikte 5 farklı şifre çözümü için kullanılacak vektörü yer kontrol istasyonuna gönderir. Gönderim esnasında düşman veya tehdit IHA tarafından bu şifreli görüntü ve 5 farklı şifre çözmek için kullanılacak anahtara ulaşılabilir. Düşman/ Tehdit IHA, bu şifreli görüntünün çözülebilmesi için 5 farklı anahtarın hangisi veya hangilerine FIFO veya LIFO algoritmaları uygulaması gerektiğini çözemez veya çözmekte zaman kaybeder. Bu sırada dost bilgiyi daha önce göndererek amacına ulaşmış olur.

Özellikle askeri alanda kullanılabilecek görüntü şifreleme yöntemi Şekil 3.50'deki biçimine benzer bir senaryoda kullanılarak verinin diğer şahıslara karşı gizliliği sağlanabilecektir.



Şekil 3.50. Örnek Şifreleme-Şifre Çözme Senaryosu

4. SONUÇ VE YORUM

Bu çalışmada, çözünürlükleri 720x720 olan uçak, 330x330 olan penguen, 256x256 olan insan ve 478x478 olan kuş temalı dört farklı görsel üzerinde iki farklı biçimde şifreleme işlemi gerçekleştirilmiştir. Elde edilen tüm şifreli görsellerin orijinal görüntüleriyle aralarında bir benzerlik söz konusu değildir.

Şifreleme yöntemlerinden ilki olan ve zaman uzayında gerçekleştirilen, rastgele permütasyon kullanılarak görüntü piksel konumlarının yer değiştirilmesi sonucunda şifreli görüntüler elde edilmiştir. Literatürde yaygın olarak kullanılan bu şifreleme veri sıkıştırma [27], sağlık alanında kişisel verilerin gizliliği konusunda [3] hizmet etmek amacıyla gerçekleştirildiği geniş bir kullanım yelpazesine sahiptir. Bu görüntülerin orijinal görüntüleriyle arasındaki histogramları incelenmiştir. Renk dağılımları, sadece piksellerin konumlarının yer değiştirmesi neticesinde tamamen aynı kalmıştır. Bu şekilde şifreli görüntü ile orijinali arasında yapılan işlemin tahmin edilebilirlik oranını da beraberinde arttırmıştır. Entropi değerlerinde ise optimum değer olan 8'e yakınsadığı için daha fazla bilgiye sahip olduğu tespit edilmiştir. Dört görsel için de aynı şekilde MSE değerleri düşük olarak elde edildiği için orijinal görsel ile şifrelendirilmiş görsel arasındaki kümülatif hata oranlarının da az olduğu tespit edilmiştir.

İkinci şifreleme yönteminde ise, çözünürlükleri 720x720 olan uçak, 330x330 olan penguen, 256x256 olan insan ve 478x478 olan kuş temalı dört farklı görüntü, frekans ve zaman uzayları kullanılarak şifreleme işlemi gerçekleştirilmiştir. Dalgacık dönüşümündeki dört farklı bileşen için ayrı ayrı rastgele permütasyon işlemi kullanılmış ve dört farklı anahtar üretilmiştir. Dört farklı şifrelenmiş bileşen birleştirilerek tekrar rastgele permütasyon işlemi uygulanmış ve beşinci anahtar oluşturularak şifrelenmiştir. Oluşturulan bu anahtarlar ise vektörlere atanmıştır. Veri yapılarında kullanılan FIFO ve LIFO algoritmaları da şifre çözülme esnasında kullanılarak şifrenin çözülmesini güçleştirmiş ve görüntünün aktarımı esnasında kolay ulaşılmasını engellemiştir. Bu şekilde şifrelenmiş görüntüler ile orijinal görüntüler arasındaki renk dağılımları histogram grafiklerinden incelendiğinde ise tamamen farklı sonuçlar elde edilmiştir. Bu sonuçlar göz önüne alındığında şifreleme işlemi hem zaman uzayında hem de frekans uzayında gerçekleştirildiği için tahmin edilebilirliği de güçleştirilmiştir.

Her iki yöntemle şifrelenmiş görüntü arasındaki çapraz korelasyon katsayısı sıfıra yakınsamakta ve iki şifreli görüntü arasındaki farklılığın büyüklüğünü kanıtlamaktadır.

Yapılan şifrelemenin kuantizasyona karşı performansının değerlendirilip hassasiyetinin ölçülmesi, bu çalışmanın temelleri üzerine kurulabilecek ve ileride yapılacak benzer çalışmalara ışık tutabilecektir.

KAYNAKLAR

[1] B. Savunma. "Bayraktar TB2: GERÇEK ZAMANLI AKTARMA VE GÖRÜNTÜ ARŞİVLEME SİSTEMİ." https://www.baykarsavunma.com/iha-15.html (accessed 2015-2019).

[2] A. D. Khalaf, "Fast Image Encryption based on Random Image Key," *International Journal of Computer Applications,* vol. 134, 2016, Art no. 3, doi: 10.13140/RG.2.1.3107.4327.

[3] K. Usman *et al.*, "Medical Image Encryption Based on Pixel Arrangement and Random Permutation for Transmission Security," presented at the 2007 9th International Conference on e-Health Networking, Application and Services, 2007.

[4] R. R. D. V.Vineeth kumar, T.Punesh Reddy, R.Sreekanth, P.Ramesh, "Designing An Efficient Image Encryption-Then Compression Via Random Permutation," *JETIR,* vol. 4, no. 04, pp. 293-296, 2017.

[5] S. Rakesh, "Image Encryption using Block Based Uniform Scrambling and Chaotic Logistic Mapping," *International Journal on Cryptography and Information Security,* vol. 2, no. 1, pp. 49-57, 2012, doi: 10.5121/ijcis.2012.2105.

[6] T. Ambritha and J. Poorani Sr, "Visual Cryptography Scheme for Colored Image using XOR with Random Key Generation," *International Journal Of Engineering And Computer Science,* 2016, doi: 10.18535/Ijecs/v5i4.45.

[7] N. Hazarika, S. Borah, and M. Saikia, "A wavelet based partial image encryption using chaotic logistic map," presented at the 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014.

[8] S. Kandar and B. C. Dhara, "Random sequence based secret sharing of an encrypted color image," presented at the 2012 1st International Conference on Recent Advances in Information Technology (RAIT), 2012.

[9] G. B. S. Lahieb Mohammed Jawad, "A Review of Color Image Encryption Techniques," *IJCSI International Journal of Computer Science Issues,* vol. 10, no. 6, pp. 266-275, 2013, Art no. 1.

[10] J. N. Abdel-Jalil, "Performance Analysis of Color Image Encryption\Decryption Techniques," *International Journal of Advanced Computer Technology (IJACT),* vol. 5, no. 4, pp. 13-17, 2016.

[11] Y. Ming, N. Bourbakis, and L. Shujun, "Data-image-video encryption," *IEEE Potentials,* vol. 23, no. 3, pp. 28-34, 2004, doi: 10.1109/mp.2004.1341784.

[12] D. C. Mishra and R. K. Sharma, "An approach for security of color image data in coordinate, geometric, and frequency domains," *Information Security Journal: A Global Perspective,* vol. 25, no. 4-6, pp. 213-234, 2016, doi: 10.1080/19393555.2016.1241323.

[13] R. Lukac and K. N. Plataniotis, *Color image processing : methods and applications* (Image processing series). Boca Raton, FL: CRC/Taylor & Francis, 2007, pp. 575 p., 26 p. of plates.

[14] S. Ramakrishnan, *Cryptographic and information security : approaches for images and videos*. Boca Raton: CRCPress, Taylor & Francis Group, CRC Press is an imprint of the Taylor & Francis Group, an informa business, 2019, pp. xxiv, 960 pages.

[15] R. C. Gonzalez and R. E. Woods, *Digital image processing*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2002, pp. xx, 793 p.

[16] E. W. Hansen, *Fourier transforms : principles and applications*. Hoboken, New Jersey: Wiley, 2014, pp. xv, 755 pages.

[17] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image processing using MATLAB*, 2nd ed. S.I.: Gatesmark Pub., 2009, pp. xviii, 826 p.

[18] T. F. Chan and J. Shen, *Image processing and analysis : variational, PDE, wavelet, and stochastic methods*. Philadelphia: Society for Industrial and Applied Mathematics, 2005, pp. xxi, 400 p.

[19] P. N. Topiwala, *Wavelet image and video compression* (The Kluwer international series in engineering and computer science, no. SECS 450). Boston: Kluwer Academic Publishers, 1998, pp. xiii, 438 p.

[20] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, NY: Pearson, 2018, pp. xvi, 1168 pages.

[21] S. A. Broughton and K. Bryan, *Discrete fourier analysis and wavelets : applications to signal and image processing*, Second edition. ed. Hoboken, NJ: Wiley, 2018, pp. xxi, 442 pages.

[22] T. B. a. Mokhtar, "The Wavelet Transform for Image Processing Applications," pp. 395-422, 2012.

[23] A. G. Pakfiliz, "Automatic of Aerial Vehicle in Cloudy Environment by Using Wavelet Enchancement Technique," *RADIOENGINEERING,* vol. 26, pp. 1169-1176, 2017, doi: 10.13164/re.2017.1169.

[24] D. S. Malik, *Data Structures Using C++*. Course Technology, 2010.

[25] Y. M. a. T. M. Francia A., "Applied Image Processing to Multimedia Information Security," *Int. Conf. Image Analysis and Signal Processing,* pp. 286-291, 2009. IEEE.

[26] X. Y. Mengmeng Guan, Weisheng Hu, "Chaotic Image Encryption Algorithm Using Frequency-Domain DNA Encoding," 2019.

[27] S. G. Ankita Vaish, Manoj Kumar, "A wavelet based approach for simultaneous compression and encryption of fused images," *Journal of King Saud University - Computer and Information Sciences,* 2017, doi: 10.1016/j.jksuci.2017.01.005.

EK 1: GÖRÜNTÜ ŞİFRELEME-ŞİFRE ÇÖZME MATLAB KODU

```
clc;
clear all;
close all;

%Load the RGB Image
imgOrg = imread('resim.jpg');
imgOrg = im2double(imgOrg);%Instead normalize the image by
255 (divide each pixel by 255) matrix, im2double will do.
sizeImg = size(imgOrg);

figure('Name','Original Image'), imshow(imgOrg); %Figure 1:
Original Image
imwrite(imgOrg,'Original Image.jpg');

%Apply Haar Wavelet on R,G,B.
[aR,hR,vR,dR] = dwt2(imgOrg(:,:,1),'haar'); %Approximate,
horizontal, vertical and diagonal of red component of image
(Haar Wavelet)
[aG,hG,vG,dG] = dwt2(imgOrg(:,:,2),'haar'); %Approximate,
horizontal, vertical and diagonal of green component of
image (Haar Wavelet)
[aB,hB,vB,dB] = dwt2(imgOrg(:,:,3),'haar'); %Approximate,
horizontal, vertical and diagonal of blue component of image
(Haar Wavelet)

%Define A,H,V,D Components for Each R,G and B matrices.
a(:,:,1) = aR; %Approximation matrix of RED Component of
image.
a(:,:,2) = aG; %Approximation matrix of GREEN Component of
image.
a(:,:,3) = aB; %Approximation matrix of BLUE Component of
image.

h(:,:,1) = hR; %Horizontal matrix of RED Component of image.
h(:,:,2) = hG; %Horizontal matrix of GREEN Component of
image.
h(:,:,3) = hB; %Horizontal matrix of BLUE Component of
image.

v(:,:,1) = vR; %Vertical matrix of RED Component of image.
v(:,:,2) = vG; %Vertical matrix of GREEN Component of image.
v(:,:,3) = vB; %Vertical matrix of BLUE Component of image.

d(:,:,1) = dR; %Diagonal matrix of RED Component of image.
d(:,:,2) = dG; %Diagonal matrix of GREEN Component of image.
d(:,:,3) = dB; %Diagonal matrix of BLUE Component of image.
```

```
figure('Name','Approximation Image'), imshow(a) ; %Figure 2:
Approximation Image
imwrite(a,'Approximation Image.jpg');
figure('Name','Horizontal Details Of Image'), imshow(h) ;
%Figure 3: Horizontal Details Of Image
imwrite(h,'Horizontal Details Of Image.jpg');
figure('Name','Vertical Details Of Image'), imshow(v) ;
%Figure 4: Vertical Details Of Image
imwrite(v,'Vertical Details Of Image.jpg');
figure('Name','Diagonal Details Of Image'), imshow(d) ;
%Figure 5: Diagonal Details Of Image
imwrite(d,'Diagonal Details Of Image.jpg');

%Encrypt-Decrypt original image then get histogram,
variance. Wavelet
%decrypted image correlation between decrypted original
image.

%Encrypt a,h,v,d respectively and then encrypt the encrypted
a,h,v,d together.
%Decrypt the encrypted a,h,v,d together. Then decrypt the
encrypted a,h,v,d respectively.

%%%%%%%%%%%%%%%%%%Encrypt original image
%Get rows, columns and colorband size
[rowsOrg, columnsOrg, numberOfColorBandsOrg] = size(imgOrg);
%Get rows, columns and colorband size of Approximation
Matrix

% Get the order to scramble them in
scrambleOrderOrg = randperm(rowsOrg*columnsOrg); %1'den
resmin pixel miktarina kadar olan sayilar gelisiguzel pixel
uzunlugundaki vektorde yer alir.
        %Burada eger rowOrg*columsOrg degeri cift ise
sutunlari FIFO tek ise LIFO olarak yolla

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
FIFO
        %algoritmasina göre ilk satirin yanina bir sonraki
satir getirilerek
        %scrambleOrder elde edilir. Ilk satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
LIFO
```

```
        %algoritmasina göre son satirin yanina bir usttteki
satir getirilerek
        %scrambleOrder elde edilir. Son satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        if mod(rowsOrg*columnsOrg,2) == 0   % Cift ise
(FIFO)
        scrambleOrderOrgSent = reshape(scrambleOrderOrg,[],
columnsOrg);
        %Receiver need to reshape like below to make
scrambleOrderOrgReceived = scrambleOrderOrg
        scrambleOrderOrgReceived =
reshape(scrambleOrderOrgSent,1, []);
        else                                % Tek ise
(LIFO), mod(rowsOrg*columnsOrg,2) == 1
        scrambleOrderOrgSent =
flip(reshape(scrambleOrderOrg,[], columnsOrg));
        %Receiver need to reshape like below
        scrambleOrderOrgReceived =
reshape(flip(scrambleOrderOrgSent),1, []);
        end

% Extract the individual red, green, and blue color
channels.
redChannelOrg = imgOrg(:,:,1);
greenChannelOrg = imgOrg(:,:,2);
blueChannelOrg = imgOrg(:,:,3);

% Scramble according to the scrambling order.
redChannelOrg = redChannelOrg(scrambleOrderOrg);
greenChannelOrg = greenChannelOrg(scrambleOrderOrg);
blueChannelOrg = blueChannelOrg(scrambleOrderOrg);

% Reshape into a 2D image
redChannelOrg = reshape(redChannelOrg, [rowsOrg,
columnsOrg]);
greenChannelOrg = reshape(greenChannelOrg, [rowsOrg,
columnsOrg]);
blueChannelOrg = reshape(blueChannelOrg, [rowsOrg,
columnsOrg]);

% Recombine separate color channels into a single, true
color RGB image.
scrambledImageOrg = cat(3, redChannelOrg, greenChannelOrg,
blueChannelOrg);

%%%%%%%%%%%%%%%%%%Decrypt original image
% Recover the image, knowing the sort order
recoverOrderOrg = zeros([rowsOrg*columnsOrg], 2);
recoverOrderOrg(:, 1) = 1 : (rowsOrg*columnsOrg);
```

```
recoverOrderOrg(:, 2) = scrambleOrderOrgReceived; %Here FIFO
or LIFO from receiver side.

% Sort this to find out where each scrambled location needs
to be sent to.
newOrderOrg = sortrows(recoverOrderOrg, 2);

% Extract just column 1, which is the order we need.
newOrderOrg = newOrderOrg(:,1);

% Unscramble according to the recoverOrder order.
redChannelOrg = redChannelOrg(newOrderOrg);
greenChannelOrg = greenChannelOrg(newOrderOrg);
blueChannelOrg = blueChannelOrg(newOrderOrg);

% Reshape into a 2D image
redChannelOrg = reshape(redChannelOrg, [rowsOrg,
columnsOrg]);
greenChannelOrg = reshape(greenChannelOrg, [rowsOrg,
columnsOrg]);
blueChannelOrg = reshape(blueChannelOrg, [rowsOrg,
columnsOrg]);

% Recombine separate color channels into a single, true
color RGB image.
rescrambledImageOrg = cat(3, redChannelOrg, greenChannelOrg,
blueChannelOrg);

%%%%%%%%%%%%%%%%%%Encrypt a,h,v,d respectively
%Get rows, columns and colorband size
[rowsA, columnsA, numberOfColorBandsA] = size(a); %Get rows,
columns and colorband size of Approximation Matrix
[rowsH, columnsH, numberOfColorBandsH] = size(h); %Get rows,
columns and colorband size of Horizontal Matrix
[rowsV, columnsV, numberOfColorBandsV] = size(v); %Get rows,
columns and colorband size of Vertical Matrix
[rowsD, columnsD, numberOfColorBandsD] = size(d); %Get rows,
columns and colorband size of Diagonal Matrix

% Get the order to scramble them in
scrambleOrderA = randperm(rowsA*columnsA); %1'den resmin
pixel miktarina kadar olan sayilar gelisiguzel pixel
uzunlugundaki vektorde yer alir.
        %Burada eger rowA*columsA degeri cift ise sutunlari
FIFO tek ise LIFO olarak yolla

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
FIFO
```

```
        %algoritmasina göre ilk satirin yanina bir sonraki
satir getirilerek
        %scrambleOrder elde edilir. Ilk satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
LIFO
        %algoritmasina göre son satirin yanina bir usttteki
satir getirilerek
        %scrambleOrder elde edilir. Son satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        if mod(rowsA*columnsA,2) == 0   % Cift ise (FIFO)
        scrambleOrderASent = reshape(scrambleOrderA,[],
columnsA);
        %Receiver need to reshape like below to make
scrambleOrderAReceived = scrambleOrderA
        scrambleOrderAReceived =
reshape(scrambleOrderASent,1, []);
        else                                % Tek ise
(LIFO), mod(rowsA*columnsA,2) == 1
        scrambleOrderASent = flip(reshape(scrambleOrderA,[],
columnsA));
        %Receiver need to reshape like below
        scrambleOrderAReceived =
reshape(flip(scrambleOrderASent),1, []);
        end
scrambleOrderH = randperm(rowsH*columnsH); %1'den resmin
pixel miktarina kadar olan sayilar gelisiguzel pixel
uzunlugundaki vektorde yer alir.
        %Burada eger rowH*columsH degeri cift ise sutunlari
FIFO tek ise LIFO olarak yolla

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
FIFO
        %algoritmasina göre ilk satirin yanina bir sonraki
satir getirilerek
        %scrambleOrder elde edilir. Ilk satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
LIFO
```

```
        %algoritmasina göre son satirin yanina bir ustteki
satir getirilerek
        %scrambleOrder elde edilir. Son satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        if mod(rowsH*columnsH,2) == 0   % Cift ise (FIFO)
        scrambleOrderHSent = reshape(scrambleOrderH,[],
columnsH);
        %Receiver need to reshape like below to make
scrambleOrderHReceived = scrambleOrderH
        scrambleOrderHReceived =
reshape(scrambleOrderHSent,1, []);
        else                                % Tek ise
(LIFO), mod(rowsH*columnsH,2) == 1
        scrambleOrderHSent = flip(reshape(scrambleOrderH,[],
columnsH));
        %Receiver need to reshape like below
        scrambleOrderHReceived =
reshape(flip(scrambleOrderHSent),1, []);
        end
scrambleOrderV = randperm(rowsV*columnsV); %1'den resmin
pixel miktarina kadar olan sayilar gelisiguzel pixel
uzunlugundaki vektorde yer alir.
        %Burada eger rowV*columsV degeri cift ise sutunlari
FIFO tek ise LIFO olarak yolla

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
FIFO
        %algoritmasina göre ilk satirin yanina bir sonraki
satir getirilerek
        %scrambleOrder elde edilir. Ilk satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
LIFO
        %algoritmasina göre son satirin yanina bir ustteki
satir getirilerek
        %scrambleOrder elde edilir. Son satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        if mod(rowsV*columnsV,2) == 0   % Cift ise (FIFO)
        scrambleOrderVSent = reshape(scrambleOrderV,[],
columnsV);
```

```
        %Receiver need to reshape like below to make
scrambleOrderVReceived = scrambleOrderV
        scrambleOrderVReceived =
reshape(scrambleOrderVSent,1, []);
        else                                % Tek ise
(LIFO), mod(rowsV*columnsV,2) == 1
        scrambleOrderVSent = flip(reshape(scrambleOrderV,[],
columnsV));
        %Receiver need to reshape like below
        scrambleOrderVReceived =
reshape(flip(scrambleOrderVSent),1, []);
        end
scrambleOrderD = randperm(rowsD*columnsD); %1'den resmin
pixel miktarina kadar olan sayilar gelisiguzel pixel
uzunlugundaki vektorde yer alir.
        %Burada eger rowD*columsD degeri cift ise sutunlari
FIFO tek ise LIFO olarak yolla

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
FIFO
        %algoritmasina göre ilk satirin yanina bir sonraki
satir getirilerek
        %scrambleOrder elde edilir. Ilk satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
LIFO
        %algoritmasina göre son satirin yanina bir ustteki
satir getirilerek
        %scrambleOrder elde edilir. Son satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        if mod(rowsD*columnsD,2) == 0   % Cift ise (FIFO)
        scrambleOrderDSent = reshape(scrambleOrderD,[],
columnsD);
        %Receiver need to reshape like below to make
scrambleOrderDReceived = scrambleOrderD
        scrambleOrderDReceived =
reshape(scrambleOrderDSent,1, []);
        else                                % Tek ise
(LIFO), mod(rowsD*columnsD,2) == 1
        scrambleOrderDSent = flip(reshape(scrambleOrderD,[],
columnsD));
        %Receiver need to reshape like below
```

```
        scrambleOrderDReceived =
reshape(flip(scrambleOrderDSent),1, []);
        end

% Extract the individual red, green, and blue color
channels.
redChannelA = aR;
greenChannelA = aG;
blueChannelA = aB;

redChannelH = hR;
greenChannelH = hG;
blueChannelH = hB;

redChannelV = vR;
greenChannelV = vG;
blueChannelV = vB;

redChannelD = dR;
greenChannelD = dG;
blueChannelD = dB;

% Scramble according to the scrambling order.
redChannelA = redChannelA(scrambleOrderA);
greenChannelA = greenChannelA(scrambleOrderA);
blueChannelA = blueChannelA(scrambleOrderA);

redChannelH = redChannelH(scrambleOrderH);
greenChannelH = greenChannelH(scrambleOrderH);
blueChannelH = blueChannelH(scrambleOrderH);

redChannelV = redChannelV(scrambleOrderV);
greenChannelV = greenChannelV(scrambleOrderV);
blueChannelV = blueChannelV(scrambleOrderV);

redChannelD = redChannelD(scrambleOrderD);
greenChannelD = greenChannelD(scrambleOrderD);
blueChannelD = blueChannelD(scrambleOrderD);

% Reshape into a 2D image
redChannelA = reshape(redChannelA, [rowsA, columnsA]);
greenChannelA = reshape(greenChannelA, [rowsA, columnsA]);
blueChannelA = reshape(blueChannelA, [rowsA, columnsA]);

redChannelH = reshape(redChannelH, [rowsH, columnsH]);
greenChannelH = reshape(greenChannelH, [rowsH, columnsH]);
blueChannelH = reshape(blueChannelH, [rowsH, columnsH]);

redChannelV = reshape(redChannelV, [rowsV, columnsV]);
greenChannelV = reshape(greenChannelV, [rowsV, columnsV]);
blueChannelV = reshape(blueChannelV, [rowsV, columnsV]);
```

```
redChannelD = reshape(redChannelD, [rowsD, columnsD]);
greenChannelD = reshape(greenChannelD, [rowsD, columnsD]);
blueChannelD = reshape(blueChannelD, [rowsD, columnsD]);

% Recombine separate color channels into a single, true
color RGB image.
scrambledImageA = cat(3, redChannelA, greenChannelA,
blueChannelA);
scrambledImageH = cat(3, redChannelH, greenChannelH,
blueChannelH);
scrambledImageV = cat(3, redChannelV, greenChannelV,
blueChannelV);
scrambledImageD = cat(3, redChannelD, greenChannelD,
blueChannelD);


    %%%%%%%%%%%%%%%%%%Encrypt a,h,v,d together
    %Concatenate 4 encrypted matrix into 1, then encrypt
altogether.
    ahvd = [scrambledImageA, scrambledImageH;
scrambledImageV, scrambledImageD];

    %Get rows, columns and colorband size
    [rowsAHVD, columnsAHVD, numberOfColorBandsAHVD] =
size(ahvd); %Get rows, columns and colorband size of ahvd
Matrix

    % Get the order to scramble them in
    scrambleOrderAHVD = randperm(rowsAHVD*columnsAHVD);
%1'den resmin pixel miktarina kadar olan sayilar gelisiguzel
pixel uzunlugundaki vektorde yer alir.

        %Burada eger rowAHVD*columsAHVD degeri cift ise
sutunlari FIFO tek ise LIFO olarak yolla

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
FIFO
        %algoritmasina göre ilk satirin yanina bir sonraki
satir getirilerek
        %scrambleOrder elde edilir. Ilk satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        %Her bir satir, sutun sayisi miktarindaki verilere
bolunerek aktarilir.
        %Bu verilerin olusturdugu matris alici tarafindan
LIFO
```

```
        %algoritmasina göre son satirin yanina bir usttteki
satir getirilerek
        %scrambleOrder elde edilir. Son satir ilk
tanimlandigi icin ilk olarak
        %cikacaktir.

        if mod(rowsAHVD*columnsAHVD,2) == 0 % Cift ise
(FIFO)
        scrambleOrderAHVDSent =
reshape(scrambleOrderAHVD,[], columnsAHVD);
        %Receiver need to reshape like below to make
scrambleOrderAHVDReceived = scrambleOrderAHVD
        scrambleOrderAHVDReceived =
reshape(scrambleOrderAHVDSent,1, []);
        else                                % Tek ise
(LIFO), mod(rowsAHVD*columnsAHVD,2) == 1
        scrambleOrderAHVDSent =
flip(reshape(scrambleOrderAHVD,[], columnsAHVD));
        %Receiver need to reshape like below
        scrambleOrderAHVDReceived =
reshape(flip(scrambleOrderAHVDSent),1, []);
        end

    % Extract the individual red, green, and blue color
channels.
    redChannelAHVD = ahvd(:,:,1);
    greenChannelAHVD = ahvd(:,:,2);
    blueChannelAHVD = ahvd(:,:,3);

    % Scramble according to the scrambling order.
    redChannelAHVD = redChannelAHVD(scrambleOrderAHVD);
    greenChannelAHVD = greenChannelAHVD(scrambleOrderAHVD);
    blueChannelAHVD = blueChannelAHVD(scrambleOrderAHVD);

    % Reshape into a 2D image
    redChannelAHVD = reshape(redChannelAHVD, [rowsAHVD,
columnsAHVD]);
    greenChannelAHVD = reshape(greenChannelAHVD, [rowsAHVD,
columnsAHVD]);
    blueChannelAHVD = reshape(blueChannelAHVD, [rowsAHVD,
columnsAHVD]);

    % Recombine separate color channels into a single, true
color RGB image.
    scrambledImageAHVD = cat(3, redChannelAHVD,
greenChannelAHVD, blueChannelAHVD);

    %%%%%%%%%%%%%%%%%%Decrypt a,h,v,d together
    % Recover the image, knowing the sort order
    recoverOrderAHVD = zeros([rowsAHVD*columnsAHVD], 2);
    recoverOrderAHVD(:, 1) = 1 : (rowsAHVD*columnsAHVD);
```

```
    recoverOrderAHVD(:, 2) = scrambleOrderAHVDReceived;
%Here FIFO or LIFO from receiver side.

    % Sort this to find out where each scrambled location
needs to be sent to.
    newOrderAHVD = sortrows(recoverOrderAHVD, 2);

    % Extract just column 1, which is the order we need.
    newOrderAHVD = newOrderAHVD(:,1);

    % Unscramble according to the recoverOrder order.
    redChannelAHVD = redChannelAHVD(newOrderAHVD);
    greenChannelAHVD = greenChannelAHVD(newOrderAHVD);
    blueChannelAHVD = blueChannelAHVD(newOrderAHVD);

    % Reshape into a 2D image
    redChannelAHVD = reshape(redChannelAHVD, [rowsAHVD,
columnsAHVD]);
    greenChannelAHVD = reshape(greenChannelAHVD, [rowsAHVD,
columnsAHVD]);
    blueChannelAHVD = reshape(blueChannelAHVD, [rowsAHVD,
columnsAHVD]);

    % Recombine separate color channels into a single, true
color RGB image.
    rescrambledImageAHVD = cat(3, redChannelAHVD,
greenChannelAHVD, blueChannelAHVD);


%%%%%%%%%%%%%%%%%%Decrypt a,h,v,d respectively
% Recover the image, knowing the sort order
recoverOrderA = zeros([rowsA*columnsA], 2);
recoverOrderA(:, 1) = 1 : (rowsA*columnsA);
recoverOrderA(:, 2) = scrambleOrderAReceived; %Here FIFO or
LIFO from receiver side.

recoverOrderH = zeros([rowsH*columnsH], 2);
recoverOrderH(:, 1) = 1 : (rowsH*columnsH);
recoverOrderH(:, 2) = scrambleOrderHReceived; %Here FIFO or
LIFO from receiver side.

recoverOrderV = zeros([rowsV*columnsV], 2);
recoverOrderV(:, 1) = 1 : (rowsV*columnsV);
recoverOrderV(:, 2) = scrambleOrderVReceived; %Here FIFO or
LIFO from receiver side.

recoverOrderD = zeros([rowsD*columnsD], 2);
recoverOrderD(:, 1) = 1 : (rowsD*columnsD);
recoverOrderD(:, 2) = scrambleOrderDReceived; %Here FIFO or
LIFO from receiver side.
```

```
% Sort this to find out where each scrambled location needs
to be sent to.
newOrderA = sortrows(recoverOrderA, 2);
newOrderH = sortrows(recoverOrderH, 2);
newOrderV = sortrows(recoverOrderV, 2);
newOrderD = sortrows(recoverOrderD, 2);

% Extract just column 1, which is the order we need.
newOrderA = newOrderA(:,1);
newOrderH = newOrderH(:,1);
newOrderV = newOrderV(:,1);
newOrderD = newOrderD(:,1);

% Unscramble according to the recoverOrder order.
redChannelA = redChannelA(newOrderA);
greenChannelA = greenChannelA(newOrderA);
blueChannelA = blueChannelA(newOrderA);

redChannelH = redChannelH(newOrderH);
greenChannelH = greenChannelH(newOrderH);
blueChannelH = blueChannelH(newOrderH);

redChannelV = redChannelV(newOrderV);
greenChannelV = greenChannelV(newOrderV);
blueChannelV = blueChannelV(newOrderV);

redChannelD = redChannelD(newOrderD);
greenChannelD = greenChannelD(newOrderD);
blueChannelD = blueChannelD(newOrderD);

% Reshape into a 2D image
redChannelA = reshape(redChannelA, [rowsA, columnsA]);
greenChannelA = reshape(greenChannelA, [rowsA, columnsA]);
blueChannelA = reshape(blueChannelA, [rowsA, columnsA]);

redChannelH = reshape(redChannelH, [rowsH, columnsH]);
greenChannelH = reshape(greenChannelH, [rowsH, columnsH]);
blueChannelH = reshape(blueChannelH, [rowsH, columnsH]);

redChannelV = reshape(redChannelV, [rowsV, columnsV]);
greenChannelV = reshape(greenChannelV, [rowsV, columnsV]);
blueChannelV = reshape(blueChannelV, [rowsV, columnsV]);

redChannelD = reshape(redChannelD, [rowsD, columnsD]);
greenChannelD = reshape(greenChannelD, [rowsD, columnsD]);
blueChannelD = reshape(blueChannelD, [rowsD, columnsD]);

% Recombine separate color channels into a single, true
color RGB image.
rescrambledImageA = cat(3, redChannelA, greenChannelA,
blueChannelA);
```

```
rescrambledImageH = cat(3, redChannelH, greenChannelH,
blueChannelH);
rescrambledImageV = cat(3, redChannelV, greenChannelV,
blueChannelV);
rescrambledImageD = cat(3, redChannelD, greenChannelD,
blueChannelD);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

figure('Name','Original Image (Encrypted)'),
imshow(scrambledImageOrg) ; %Figure 6: Original Image
(Encrypted)
imwrite(scrambledImageOrg,'Original Image (Encrypted).jpg');

figure('Name','Approximation Image (Encrypted)'),
imshow(scrambledImageA) ; %Figure 7: Approximation Image
(Encrypted)
imwrite(scrambledImageA,'Approximation Image
(Encrypted).jpg');
figure('Name','Horizontal Details Of Image (Encrypted)'),
imshow(scrambledImageH) ; %Figure 8: Horizontal Details Of
Image (Encrypted)
imwrite(scrambledImageH,'Horizontal Details Of Image
(Encrypted).jpg');
figure('Name','Vertical Details Of Image (Encrypted)'),
imshow(scrambledImageV) ; %Figure 9: Vertical Details Of
Image (Encrypted)
imwrite(scrambledImageV,'Vertical Details Of Image
(Encrypted).jpg');
figure('Name','Diagonal Details Of Image (Encrypted)'),
imshow(scrambledImageD) ; %Figure 10: Diagonal Details Of
Image (Encrypted)
imwrite(scrambledImageD,'Diagonal Details Of Image
(Encrypted).jpg');


figure('Name','AHVD Image'), imshow(ahvd) ; %Figure 11: AHVD
Image
imwrite(ahvd,'AHVD Image.jpg');

figure('Name','AHVD Image (Encrypted)'),
imshow(scrambledImageAHVD) ; %Figure 12: AHVD Image
(Encrypted)
imwrite(scrambledImageAHVD,'AHVD Image (Encrypted).jpg');

figure('Name','Original Image (Decrypted)'),
imshow(rescrambledImageOrg) ; %Figure 13: Original Image
(Decrypted)
imwrite(rescrambledImageOrg,'Original Image
(Decrypted).jpg');
```

```
figure('Name','Approximation Image (Decrypted)'),
imshow(rescrambledImageA) ; %Figure 14: Approximation Image
(Decrypted)
imwrite(rescrambledImageA,'Approximation Image
(Decrypted).jpg');
figure('Name','Horizontal Details Of Image (Decrypted)'),
imshow(rescrambledImageH) ; %Figure 15: Horizontal Details
Of Image (Decrypted)
imwrite(rescrambledImageH,'Horizontal Details Of Image
(Decrypted).jpg');
figure('Name','Vertical Details Of Image (Decrypted)'),
imshow(rescrambledImageV) ; %Figure 16: Vertical Details Of
Image (Decrypted)
imwrite(rescrambledImageV,'Vertical Details Of Image
(Decrypted).jpg');
figure('Name','Diagonal Details Of Image (Decrypted)'),
imshow(rescrambledImageD) ; %Figure 17: Diagonal Details Of
Image (Decrypted)
imwrite(rescrambledImageD,'Diagonal Details Of Image
(Decrypted).jpg');

figure('Name','AHVD Image (Decrypted)'),
imshow(rescrambledImageAHVD) ; %Figure 18: AHVD Image
(Decrypted)
imwrite(rescrambledImageAHVD,'AHVD Image (Decrypted).jpg');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

fig19 = figure('Name','Histogram Of Original Image');
%Figure 19: Histogram Of Original Image
    plot(imhist(imgOrg(:,:,1)),'r')
    hold on,
    plot(imhist(imgOrg(:,:,2)),'g')
    plot(imhist(imgOrg(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig19,'Histogram Of Original Image.jpg')


fig20 = figure('Name','Histogram Of AHVD Image'); %Figure
20: Histogram Of AHVD Image
    plot(imhist(ahvd(:,:,1)),'r')
    hold on,
    plot(imhist(ahvd(:,:,2)),'g')
    plot(imhist(ahvd(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig20,'Histogram Of AHVD Image.jpg')
```

```
fig21 = figure('Name','Histogram Of Approximation Image');
%Figure 21: Histogram Of Approximation Image
    plot(imhist(a(:,:,1)),'r')
    hold on,
    plot(imhist(a(:,:,2)),'g')
    plot(imhist(a(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig21,'Histogram Of Approximation Image.jpg')

fig22 = figure('Name','Histogram Of Horizontal Image');
%Figure 22: Histogram Of Horizontal Image
    plot(imhist(h(:,:,1)),'r')
    hold on,
    plot(imhist(h(:,:,2)),'g')
    plot(imhist(h(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig22,'Histogram Of Horizontal Image.jpg')

fig23 = figure('Name','Histogram Of Vertical Image');
%Figure 23: Histogram Of Vertical Image
    plot(imhist(v(:,:,1)),'r')
    hold on,
    plot(imhist(v(:,:,2)),'g')
    plot(imhist(v(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig23,'Histogram Of Vertical Image.jpg')

fig24 = figure('Name','Histogram Of Diagonal Image');
%Figure 24: Histogram Of Diagonal Image
    plot(imhist(d(:,:,1)),'r')
    hold on,
    plot(imhist(d(:,:,2)),'g')
    plot(imhist(d(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig24,'Histogram Of Diagonal Image.jpg')



fig25 = figure('Name','Histogram Of Original Image
(Encrypted)'); %Figure 25: Histogram Of Original Image
(Encrypted)
    plot(imhist(scrambledImageOrg(:,:,1)),'r')
    hold on,
    plot(imhist(scrambledImageOrg(:,:,2)),'g')
    plot(imhist(scrambledImageOrg(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
```

```
    saveas(fig25,'Histogram Of Original Image
(Encrypted).jpg')


fig26 = figure('Name','Histogram Of Approximation Image
(Encrypted)'); %Figure 26: Histogram Of Approximation Image
(Encrypted)
    plot(imhist(scrambledImageA(:,:,1)),'r')
    hold on,
    plot(imhist(scrambledImageA(:,:,2)),'g')
    plot(imhist(scrambledImageA(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig26,'Histogram Of Approximation Image
(Encrypted).jpg')
fig27 = figure('Name','Histogram Of Horizontal Details Of
Image (Encrypted)'); %Figure 27: Histogram Of Horizontal
Details Of Image (Encrypted)
    plot(imhist(scrambledImageH(:,:,1)),'r')
    hold on,
    plot(imhist(scrambledImageH(:,:,2)),'g')
    plot(imhist(scrambledImageH(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig27,'Histogram Of Horizontal Details Of Image
(Encrypted).jpg')
fig28 = figure('Name','Histogram Of Vertical Details Of
Image (Encrypted)'); %Figure 28: Histogram Of Vertical
Details Of Image (Encrypted)
    plot(imhist(scrambledImageV(:,:,1)),'r')
    hold on,
    plot(imhist(scrambledImageV(:,:,2)),'g')
    plot(imhist(scrambledImageV(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig28,'Histogram Of Vertical Details Of Image
(Encrypted).jpg')
fig29 = figure('Name','Histogram Of Diagonal Details Of
Image (Encrypted)'); %Figure 29: Histogram Of Diagonal
Details Of Image (Encrypted)
    plot(imhist(scrambledImageD(:,:,1)),'r')
    hold on,
    plot(imhist(scrambledImageD(:,:,2)),'g')
    plot(imhist(scrambledImageD(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig29,'Histogram Of Diagonal Details Of Image
(Encrypted).jpg')
```

```
fig30 = figure('Name','Histogram Of AHVD Image
(Encrypted)'); %Figure 30: Histogram Of AHVD Image
(Encrypted)
    plot(imhist(scrambledImageAHVD(:,:,1)),'r')
    hold on,
    plot(imhist(scrambledImageAHVD(:,:,2)),'g')
    plot(imhist(scrambledImageAHVD(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig30,'Histogram Of AHVD Image (Encrypted).jpg')

fig31 = figure('Name','Histogram Of Original Image
(Decrypted)'); %Figure 31: Histogram Of Original Image
(Decrypted)
    plot(imhist(rescrambledImageOrg(:,:,1)),'r')
    hold on,
    plot(imhist(rescrambledImageOrg(:,:,2)),'g')
    plot(imhist(rescrambledImageOrg(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig31,'Histogram Of Original Image
(Decrypted).jpg')

fig32 = figure('Name','Histogram Of Approximation Image
(Decrypted)'); %Figure 32: Histogram Of Approximation Image
(Decrypted)
    plot(imhist(rescrambledImageA(:,:,1)),'r')
    hold on,
    plot(imhist(rescrambledImageA(:,:,2)),'g')
    plot(imhist(rescrambledImageA(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig32,'Histogram Of Approximation Image
(Decrypted).jpg')
fig33 = figure('Name','Histogram Of Horizontal Details Of
Image (Decrypted)'); %Figure 33: Histogram Of Horizontal
Details Of Image (Decrypted)
    plot(imhist(rescrambledImageH(:,:,1)),'r')
    hold on,
    plot(imhist(rescrambledImageH(:,:,2)),'g')
    plot(imhist(rescrambledImageH(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig33,'Histogram Of Horizontal Details Of Image
(Decrypted).jpg')
fig34 = figure('Name','Histogram Of Vertical Details Of
Image (Decrypted)'); %Figure 34: Histogram Of Vertical
Details Of Image (Decrypted)
    plot(imhist(rescrambledImageV(:,:,1)),'r')
    hold on,
    plot(imhist(rescrambledImageV(:,:,2)),'g')
```

```
    plot(imhist(rescrambledImageV(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig34,'Histogram Of Vertical Details Of Image
(Decrypted).jpg')
fig35 = figure('Name','Histogram Of Diagonal Details Of
Image (Decrypted)'); %Figure 35: Histogram Of Diagonal
Details Of Image (Decrypted)
    plot(imhist(rescrambledImageD(:,:,1)),'r')
    hold on,
    plot(imhist(rescrambledImageD(:,:,2)),'g')
    plot(imhist(rescrambledImageD(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig35,'Histogram Of Diagonal Details Of Image
(Decrypted).jpg')

fig36 = figure('Name','Histogram Of AHVD Image
(Decrypted)'); %Figure 36: Histogram Of AHVD Image
(Decrypted)
    plot(imhist(rescrambledImageAHVD(:,:,1)),'r')
    hold on,
    plot(imhist(rescrambledImageAHVD(:,:,2)),'g')
    plot(imhist(rescrambledImageAHVD(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig36,'Histogram Of AHVD Image (Decrypted).jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

fig37 = figure('Name','Variance Of Original Image');
text(0.5, 0.5, num2str(var(imgOrg(:)))); %Figure 37:
Variance Of Original Image
saveas(fig37,'Variance Of Original Image.jpg')

fig38 = figure('Name','Variance Of Original Image
(Encrypted)'); text(0.5, 0.5,
num2str(var(scrambledImageOrg(:)))); %Figure 38: Variance Of
Original Image (Encrypted)
saveas(fig38,'Variance Of Original Image (Encrypted).jpg')


fig39 = figure('Name','Variance Of AHVD Image'); text(0.5,
0.5, num2str(var(ahvd(:)))); %Figure 39: Variance Of AHVD
Image
saveas(fig39,'Variance Of AHVD Image.jpg');

fig40 = figure('Name','Variance Of Approximation Image');
text(0.5, 0.5, num2str(var(a(:)))); %Figure 40: Variance Of
Approximation Image
```

```
saveas(fig40,'Variance Of Approximation Image.jpg');

fig41 = figure('Name','Variance Of Horizontal Details Of
Image'); text(0.5, 0.5, num2str(var(h(:)))); %Figure 41:
Variance Of Horizontal Details Of Image
saveas(fig41,'Variance Of Horizontal Details Of Image.jpg');

fig42 = figure('Name','Variance Of Vertical Details Of
Image'); text(0.5, 0.5, num2str(var(v(:)))); %Figure 42:
Variance Of Vertical Details Of Image
saveas(fig42,'Variance Of Vertical Details Of Image.jpg');

fig43 = figure('Name','Variance Of Diagonal Details Of
Image'); text(0.5, 0.5, num2str(var(d(:)))); %Figure 43:
Variance Of Diagonal Details Of Image
saveas(fig43,'Variance Of Diagonal Details Of Image.jpg');



fig44 = figure('Name','Variance Of Approximation Image
(Encrypted)'); text(0.5, 0.5,
num2str(var(scrambledImageA(:)))); %Figure 44: Variance Of
Approximation Image (Encrypted)
saveas(fig44,'Variance Of Approximation Image
(Encrypted).jpg')
fig45 = figure('Name','Variance Of Horizontal Details Of
Image (Encrypted)'); text(0.5, 0.5,
num2str(var(scrambledImageH(:)))); %Figure 45: Variance Of
Horizontal Details Of Image (Encrypted)
saveas(fig45,'Variance Of Horizontal Details Of Image
(Encrypted).jpg')
fig46 = figure('Name','Variance Of Vertical Details Of Image
(Encrypted)'); text(0.5, 0.5,
num2str(var(scrambledImageV(:)))); %Figure 46: Variance Of
Vertical Details Of Image (Encrypted)
saveas(fig46,'Variance Of Vertical Details Of Image
(Encrypted).jpg')
fig47 = figure('Name','Variance Of Diagonal Details Of Image
(Encrypted)'); text(0.5, 0.5,
num2str(var(scrambledImageD(:)))); %Figure 47: Variance Of
Diagonal Details Of Image (Encrypted)
saveas(fig47,'Variance Of Diagonal Details Of Image
(Encrypted).jpg')

fig48 = figure('Name','Variance Of AHVD Image (Encrypted)');
text(0.5, 0.5, num2str(var(scrambledImageAHVD(:)))); %Figure
48: Variance Of AHVD Image (Encrypted)
saveas(fig48,'Variance Of AHVD Image (Encrypted).jpg')

fig49 = figure('Name','Variance Of Original Image
(Decrypted)'); text(0.5, 0.5,
```

```
num2str(var(rescrambledImageOrg(:)))); %Figure 49: Variance
Of Original Image (Decrypted)
saveas(fig49,'Variance Of Original Image (Decrypted).jpg')

fig50 = figure('Name','Variance Of Approximation Image
(Decrypted)'); text(0.5, 0.5,
num2str(var(rescrambledImageA(:)))); %Figure 50: Variance Of
Approximation Image (Decrypted)
saveas(fig50,'Variance Of Approximation Image
(Decrypted).jpg')
fig51 = figure('Name','Variance Of Horizontal Details Of
Image (Decrypted)'); text(0.5, 0.5,
num2str(var(rescrambledImageH(:)))); %Figure 51: Variance Of
Horizontal Details Of Image (Decrypted)
saveas(fig51,'Variance Of Horizontal Details Of Image
(Decrypted).jpg')
fig52 = figure('Name','Variance Of Vertical Details Of Image
(Decrypted)'); text(0.5, 0.5,
num2str(var(rescrambledImageV(:)))); %Figure 52: Variance Of
Vertical Details Of Image (Decrypted)
saveas(fig52,'Variance Of Vertical Details Of Image
(Decrypted).jpg')
fig53 = figure('Name','Variance Of Diagonal Details Of Image
(Decrypted)'); text(0.5, 0.5,
num2str(var(rescrambledImageD(:)))); %Figure 53: Variance Of
Diagonal Details Of Image (Decrypted)
saveas(fig53,'Variance Of Diagonal Details Of Image
(Decrypted).jpg')

fig54 = figure('Name','Variance Of AHVD Image (Decrypted)');
text(0.5, 0.5, num2str(var(rescrambledImageAHVD(:))));
%Figure 54: Variance Of AHVD Image (Decrypted)
saveas(fig54,'Variance Of AHVD Image (Decrypted).jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

fig55 = figure('Name','Standard Deviation Of Original
Image'); text(0.5, 0.5, num2str(std2(imgOrg))); %Figure 55:
Standard Deviation Of Original Image
saveas(fig55,'Standard Deviation Of Original Image.jpg')


fig56 = figure('Name','Standard Deviation Of AHVD Image');
text(0.5, 0.5, num2str(std2(ahvd))); %Figure 56: Standard
Deviation Of AHVD Image
saveas(fig56,'Standard Deviation Of AHVD Image.jpg');

fig57 = figure('Name','Standard Deviation Of Approximation
Image'); text(0.5, 0.5, num2str(std2(a))); %Figure 57:
Standard Deviation Of Approximation Image
```

```
saveas(fig57,'Standard Deviation Of Approximation
Image.jpg');

fig58 = figure('Name','Standard Deviation Of Horizontal
Details Of Image'); text(0.5, 0.5, num2str(std2(h)));
%Figure 58: Standard Deviation Of Horizontal Details Of
Image
saveas(fig58,'Standard Deviation Of Horizontal Details Of
Image.jpg');

fig59 = figure('Name','Standard Deviation Of Vertical
Details Of Image'); text(0.5, 0.5, num2str(std2(v)));
%Figure 59: Standard Deviation Of Vertical Details Of Image
saveas(fig59,'Standard Deviation Of Vertical Details Of
Image.jpg');

fig60 = figure('Name','Standard Deviation Of Diagonal
Details Of Image'); text(0.5, 0.5, num2str(std2(d)));
%Figure 60: Standard Deviation Of Diagonal Details Of Image
saveas(fig60,'Standard Deviation Of Diagonal Details Of
Image.jpg');


fig61 = figure('Name','Standard Deviation Of Original Image
(Encrypted)'); text(0.5, 0.5,
num2str(std2(scrambledImageOrg))); %Figure 61: Standard
Deviation Of Original Image (Encrypted)
saveas(fig61,'Standard Deviation Of Original Image
(Encrypted).jpg')

fig62 = figure('Name','Standard Deviation Of Approximation
Image (Encrypted)'); text(0.5, 0.5,
num2str(std2(scrambledImageA))); %Figure 62: Standard
Deviation Of Approximation Image (Encrypted)
saveas(fig62,'Standard Deviation Of Approximation Image
(Encrypted).jpg')
fig63 = figure('Name','Standard Deviation Of Horizontal
Details Of Image (Encrypted)'); text(0.5, 0.5,
num2str(std2(scrambledImageH))); %Figure 63: Standard
Deviation Of Horizontal Details Of Image (Encrypted)
saveas(fig63,'Standard Deviation Of Horizontal Details Of
Image (Encrypted).jpg')
fig64 = figure('Name','Standard Deviation Of Vertical
Details Of Image (Encrypted)'); text(0.5, 0.5,
num2str(std2(scrambledImageV))); %Figure 64: Standard
Deviation Of Vertical Details Of Image (Encrypted)
saveas(fig64,'Standard Deviation Of Vertical Details Of
Image (Encrypted).jpg')
fig65 = figure('Name','Standard Deviation Of Diagonal
Details Of Image (Encrypted)'); text(0.5, 0.5,
```

```
num2str(std2(scrambledImageD))); %Figure 65: Standard
Deviation Of Diagonal Details Of Image (Encrypted)
saveas(fig65,'Standard Deviation Of Diagonal Details Of
Image (Encrypted).jpg')

fig66 = figure('Name','Standard Deviation Of AHVD Image
(Encrypted)'); text(0.5, 0.5,
num2str(std2(scrambledImageAHVD))); %Figure 66: Standard
Deviation Of AHVD Image (Encrypted)
saveas(fig66,'Standard Deviation Of AHVD Image
(Encrypted).jpg')

fig67 = figure('Name','Standard Deviation Of Original Image
(Decrypted)'); text(0.5, 0.5,
num2str(std2(rescrambledImageOrg))); %Figure 67: Standard
Deviation Of Original Image (Decrypted)
saveas(fig67,'Standard Deviation Of Original Image
(Decrypted).jpg')

fig68 = figure('Name','Standard Deviation Of Approximation
Image (Decrypted)'); text(0.5, 0.5,
num2str(std2(rescrambledImageA(:)))); %Figure 68: Standard
Deviation Of Approximation Image (Decrypted)
saveas(fig68,'Standard Deviation Of Approximation Image
(Decrypted).jpg')
fig69 = figure('Name','Standard Deviation Of Horizontal
Details Of Image (Decrypted)'); text(0.5, 0.5,
num2str(std2(rescrambledImageH))); %Figure 69: Standard
Deviation Of Horizontal Details Of Image (Decrypted)
saveas(fig69,'Standard Deviation Of Horizontal Details Of
Image (Decrypted).jpg')
fig70 = figure('Name','Standard Deviation Of Vertical
Details Of Image (Decrypted)'); text(0.5, 0.5,
num2str(std2(rescrambledImageV))); %Figure 70: Standard
Deviation Of Vertical Details Of Image (Decrypted)
saveas(fig70,'Standard Deviation Of Vertical Details Of
Image (Decrypted).jpg')
fig71 = figure('Name','Standard Deviation Of Diagonal
Details Of Image (Decrypted)'); text(0.5, 0.5,
num2str(std2(rescrambledImageD))); %Figure 71: Standard
Deviation Of Diagonal Details Of Image (Decrypted)
saveas(fig71,'Standard Deviation Of Diagonal Details Of
Image (Decrypted).jpg')

fig72 = figure('Name','Standard Deviation Of AHVD Image
(Decrypted)'); text(0.5, 0.5,
num2str(std2(rescrambledImageAHVD))); %Figure 72: Standard
Deviation Of AHVD Image (Decrypted)
saveas(fig72,'Standard Deviation Of AHVD Image
(Decrypted).jpg')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Inverse Discrete Wavelet Transform of R,G and B matrices.
invR =
idwt2(redChannelA,redChannelH,redChannelV,redChannelD,'haar'
,sizeImg); %Inverse Discrete Wavelet Transforn for RED
Component of Image
invG =
idwt2(greenChannelA,greenChannelH,greenChannelV,greenChannel
D,'haar',sizeImg); %Inverse Discrete Wavelet Transforn for
GREEN Component of Image
invB =
idwt2(blueChannelA,blueChannelH,blueChannelV,blueChannelD,'h
aar',sizeImg); %Inverse Discrete Wavelet Transforn for BLUE
Component of Image

newImg = cat(3, invR, invG, invB); %Concatenate R,G and B
matrices to get RGB Image.

figure('Name','After Inverse Wavelet Image'),
imshow(newImg); %Figure 73: After Inverse Wavelet Image
imwrite(newImg,'After Inverse Wavelet Image.jpg');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fig74 = figure('Name','Histogram Of After Inverse Wavelet
Image'); %Figure 74: Histogram Of After Inverse Wavelet
Image
    plot(imhist(newImg(:,:,1)),'r')
    hold on,
    plot(imhist(newImg(:,:,2)),'g')
    plot(imhist(newImg(:,:,3)),'b'),
    legend('Red channel','Green channel','Blue channel');
    hold off,
    saveas(fig74,'Histogram Of After Inverse Wavelet
Image.jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fig75 = figure('Name','Standard Deviation Of After Inverse
Wavelet Image'); text(0.5, 0.5, num2str(std2(newImg)));
%Figure 75: Standard Deviation Of After Inverse Wavelet
Image
saveas(fig75,'Standard Deviation Of After Inverse Wavelet
Image.jpg')
```

```
fig76 = figure('Name','Variance Of After Inverse Wavelet
Image'); text(0.5, 0.5, num2str(var(newImg(:))));  %Figure
76: Variance Of After Inverse Wavelet Image
saveas(fig76,'Variance Of After Inverse Wavelet Image.jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

fig77 = figure('Name','Correlation Between Encrypted
Original Image - Encrypted AHVD Image (RED CHANNEL)');
text(0.5, 0.5, num2str(corr2(scrambledImageOrg(:,:,1),
scrambledImageAHVD(:,:,1))));  %Figure 77: Correlation
Between Encrypted Original Image - Encrypted AHVD Image (RED
CHANNEL)
saveas(fig77,'Correlation Between Encrypted Original Image -
Encrypted AHVD Image (RED CHANNEL).jpg')
fig78 = figure('Name','Correlation Between Encrypted
Original Image - Encrypted AHVD Image (GREEN CHANNEL)');
text(0.5, 0.5, num2str(corr2(scrambledImageOrg(:,:,2),
scrambledImageAHVD(:,:,2))));  %Figure 78: Correlation
Between Encrypted Original Image - Encrypted AHVD Image
(GREEN CHANNEL)
saveas(fig78,'Correlation Between Encrypted Original Image -
Encrypted AHVD Image (GREEN CHANNEL).jpg')
fig79 = figure('Name','Correlation Between Encrypted
Original Image - Encrypted AHVD Image (BLUE CHANNEL)');
text(0.5, 0.5, num2str(corr2(scrambledImageOrg(:,:,3),
scrambledImageAHVD(:,:,3))));  %Figure 79: Correlation
Between Encrypted Original Image - Encrypted AHVD Image
(BLUE CHANNEL)
saveas(fig79,'Correlation Between Encrypted Original Image -
Encrypted AHVD Image (BLUE CHANNEL).jpg')
fig80 = figure('Name','Correlation Between Encrypted
Original Image - Encrypted AHVD Image (GRAY SCALE)');
text(0.5, 0.5, num2str(corr2(rgb2gray(scrambledImageOrg),
rgb2gray(scrambledImageAHVD))));  %Figure 80: Correlation
Between Encrypted Original Image - Encrypted AHVD Image
(GRAY SCALE)
saveas(fig80,'Correlation Between Encrypted Original Image -
Encrypted AHVD Image (GRAY SCALE).jpg')

fig81 = figure('Name','Correlation Between Decrypted
Original Image - Fully Decrypted AHVD Image (RED CHANNEL)');
text(0.5, 0.5, num2str(corr2(rescrambledImageOrg(:,:,1),
newImg(:,:,1))));  %Figure 81: Correlation Between Decrypted
Original Image - Fully Decrypted AHVD Image (RED CHANNEL)
saveas(fig81,'Correlation Between Decrypted Original Image -
Fully Decrypted AHVD Image (RED CHANNEL).jpg')
fig82 = figure('Name','Correlation Between Decrypted
Original Image - Fully Decrypted AHVD Image (GREEN
CHANNEL)'); text(0.5, 0.5,
```

```
num2str(corr2(rescrambledImageOrg(:,:,2), newImg(:,:,2))));
%Figure 82: Correlation Between Decrypted Original Image -
Fully Decrypted AHVD Image (GREEN CHANNEL)
saveas(fig82,'Correlation Between Decrypted Original Image -
Fully Decrypted AHVD Image (GREEN CHANNEL).jpg')
fig83 = figure('Name','Correlation Between Decrypted
Original Image - Fully Decrypted AHVD Image (BLUE
CHANNEL)'); text(0.5, 0.5,
num2str(corr2(rescrambledImageOrg(:,:,3), newImg(:,:,3))));
%Figure 83: Correlation Between Decrypted Original Image -
Fully Decrypted AHVD Image (BLUE CHANNEL)
saveas(fig83,'Correlation Between Decrypted Original Image -
Fully Decrypted AHVD Image (BLUE CHANNEL).jpg')
fig84 = figure('Name','Correlation Between Decrypted
Original Image - Fully Decrypted AHVD Image (GRAY SCALE)');
text(0.5, 0.5, num2str(corr2(rgb2gray(rescrambledImageOrg),
rgb2gray(newImg)))); %Figure 84: Correlation Between
Decrypted Original Image - Fully Decrypted AHVD Image (GRAY
SCALE)
saveas(fig84,'Correlation Between Decrypted Original Image -
Fully Decrypted AHVD Image (GRAY SCALE).jpg')


fig85 = figure('Name','Correlation Between Original Image -
Encrypted AHVD Image (RED CHANNEL)'); text(0.5, 0.5,
num2str(corr2(imgOrg(:,:,1), scrambledImageAHVD(:,:,1))));
%Figure 85: Correlation Between Original Image - Encrypted
AHVD Image (RED CHANNEL)
saveas(fig85,'Correlation Between Original Image - Encrypted
AHVD Image (RED CHANNEL).jpg')
fig86 = figure('Name','Correlation Between Original Image -
Encrypted AHVD Image (GREEN CHANNEL)'); text(0.5, 0.5,
num2str(corr2(imgOrg(:,:,2), scrambledImageAHVD(:,:,2))));
%Figure 86: Correlation Between Original Image - Encrypted
AHVD Image (GREEN CHANNEL)
saveas(fig86,'Correlation Between Original Image - Encrypted
AHVD Image (GREEN CHANNEL).jpg')
fig87 = figure('Name','Correlation Between Original Image -
Encrypted AHVD Image (BLUE CHANNEL)'); text(0.5, 0.5,
num2str(corr2(imgOrg(:,:,3), scrambledImageAHVD(:,:,3))));
%Figure 87: Correlation Between Original Image - Encrypted
AHVD Image (BLUE CHANNEL)
saveas(fig87,'Correlation Between Original Image - Encrypted
AHVD Image (BLUE CHANNEL).jpg')
fig88 = figure('Name','Correlation Between Original Image -
Encrypted AHVD Image (GRAY SCALE)'); text(0.5, 0.5,
num2str(corr2(rgb2gray(imgOrg),
rgb2gray(scrambledImageAHVD)))); %Figure 88: Correlation
Between Original Image - Encrypted AHVD Image (GRAY SCALE)
saveas(fig88,'Correlation Between Original Image - Encrypted
AHVD Image (GRAY SCALE).jpg')
```

```
fig89 = figure('Name','Correlation Between Original Image - Encrypted Original Image (RED CHANNEL)'); text(0.5, 0.5, num2str(corr2(imgOrg(:,:,1), scrambledImageOrg(:,:,1))));
%Figure 89: Correlation Between Original Image - Encrypted Original Image (RED CHANNEL)
saveas(fig89,'Correlation Between Original Image - Encrypted Original Image (RED CHANNEL).jpg')
fig90 = figure('Name','Correlation Between Original Image - Encrypted Original Image (GREEN CHANNEL)'); text(0.5, 0.5, num2str(corr2(imgOrg(:,:,2), scrambledImageOrg(:,:,2))));
%Figure 90: Correlation Between Original Image - Encrypted Original Image (GREEN CHANNEL)
saveas(fig90,'Correlation Between Original Image - Encrypted Original Image (GREEN CHANNEL).jpg')
fig91 = figure('Name','Correlation Between Original Image - Encrypted Original Image (BLUE CHANNEL)'); text(0.5, 0.5, num2str(corr2(imgOrg(:,:,3), scrambledImageOrg(:,:,3))));
%Figure 91: Correlation Between Original Image - Encrypted Original Image (BLUE CHANNEL)
saveas(fig91,'Correlation Between Original Image - Encrypted Original Image (BLUE CHANNEL).jpg')
fig92 = figure('Name','Correlation Between Original Image - Encrypted Original Image (GRAY SCALE)'); text(0.5, 0.5, num2str(corr2(rgb2gray(imgOrg), rgb2gray(scrambledImageOrg)))); %Figure 92: Correlation Between Original Image - Encrypted Original Image (GRAY SCALE)
saveas(fig92,'Correlation Between Original Image - Encrypted Original Image (GRAY SCALE).jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

squaredErrorImage_R_OrgScrOrg = (double(imgOrg(:,:,1)) - double(scrambledImageOrg(:,:,1))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a single number).
mse_R_OrgScrOrg = sum(sum(squaredErrorImage_R_OrgScrOrg)) / (rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE according to the formula.
PSNR_R_OrgScrOrg = 10 * log10( 256^2 / mse_R_OrgScrOrg);

squaredErrorImage_G_OrgScrOrg = (double(imgOrg(:,:,2)) - double(scrambledImageOrg(:,:,2))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a single number).
```

```
mse_G_OrgScrOrg = sum(sum(squaredErrorImage_G_OrgScrOrg)) /
(rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
PSNR_G_OrgScrOrg = 10 * log10( 256^2 / mse_G_OrgScrOrg);

squaredErrorImage_B_OrgScrOrg = (double(imgOrg(:,:,3)) -
double(scrambledImageOrg(:,:,3))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a
single number).
mse_B_OrgScrOrg = sum(sum(squaredErrorImage_B_OrgScrOrg)) /
(rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
PSNR_B_OrgScrOrg = 10 * log10( 256^2 / mse_B_OrgScrOrg);

squaredErrorImage_OrgScrOrg = (double(rgb2gray(imgOrg)) -
double(rgb2gray(scrambledImageOrg))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a
single number).
mse_OrgScrOrg = sum(sum(squaredErrorImage_OrgScrOrg)) /
(rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
PSNR_OrgScrOrg = 10 * log10( 256^2 / mse_OrgScrOrg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

squaredErrorImage_R_OrgAhvd = (double(imgOrg(:,:,1)) -
double(scrambledImageAHVD(:,:,1))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a
single number).
mse_R_OrgAhvd = sum(sum(squaredErrorImage_R_OrgAhvd)) /
(rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
PSNR_R_OrgAhvd = 10 * log10( 256^2 / mse_R_OrgAhvd);

squaredErrorImage_G_OrgAhvd = (double(imgOrg(:,:,2)) -
double(scrambledImageAHVD(:,:,2))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a
single number).
mse_G_OrgAhvd = sum(sum(squaredErrorImage_G_OrgAhvd)) /
(rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
```

```
PSNR_G_OrgAhvd = 10 * log10( 256^2 / mse_G_OrgAhvd);

squaredErrorImage_B_OrgAhvd = (double(imgOrg(:,:,3)) -
double(scrambledImageAHVD(:,:,3))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a
single number).
mse_B_OrgAhvd = sum(sum(squaredErrorImage_B_OrgAhvd)) /
(rowsOrg * columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
PSNR_B_OrgAhvd = 10 * log10( 256^2 / mse_B_OrgAhvd);

squaredErrorImage_OrgAhvd = (double(rgb2gray(imgOrg)) -
double(rgb2gray(scrambledImageAHVD))) .^ 2;
% Sum the Squared Image and divide by the number of elements
% to get the Mean Squared Error.  It will be a scalar (a
single number).
mse_OrgAhvd = sum(sum(squaredErrorImage_OrgAhvd)) / (rowsOrg
* columnsOrg);
% Calculate PSNR (Peak Signal to Noise Ratio) from the MSE
according to the formula.
PSNR_OrgAhvd = 10 * log10( 256^2 / mse_OrgAhvd);

fig93 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted AHVD Image (RED CHANNEL)');
text(0.5, 0.5, num2str(mse_R_OrgAhvd)); %Figure 93: Mean
Square Error (MSE) Between Original Image - Encrypted AHVD
Image (RED CHANNEL)
saveas(fig93,'Mean Square Error (MSE) Between Original Image
- Encrypted AHVD Image (RED CHANNEL).jpg')

fig94 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted AHVD Image (RED CHANNEL)');
text(0.5, 0.5, num2str(PSNR_R_OrgAhvd)); %Figure 94: Peak To
Noise Ratio (PSNR) Between Original Image - Encrypted AHVD
Image (RED CHANNEL)
saveas(fig94,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted AHVD Image (RED CHANNEL).jpg')


fig95 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted AHVD Image (GREEN CHANNEL)');
text(0.5, 0.5, num2str(mse_G_OrgAhvd)); %Figure 95: Mean
Square Error (MSE) Between Original Image - Encrypted AHVD
Image (GREEN CHANNEL)
saveas(fig95,'Mean Square Error (MSE) Between Original Image
- Encrypted AHVD Image (GREEN CHANNEL).jpg')

fig96 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted AHVD Image (GREEN CHANNEL)');
```

```
text(0.5, 0.5, num2str(PSNR_G_OrgAhvd)); %Figure 96: Peak To
Noise Ratio (PSNR) Between Original Image - Encrypted AHVD
Image (GREEN CHANNEL)
saveas(fig96,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted AHVD Image (GREEN CHANNEL).jpg')


fig97 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted AHVD Image (BLUE CHANNEL)');
text(0.5, 0.5, num2str(mse_B_OrgAhvd)); %Figure 97: Mean
Square Error (MSE) Between Original Image - Encrypted AHVD
Image (BLUE CHANNEL)
saveas(fig97,'Mean Square Error (MSE) Between Original Image
- Encrypted AHVD Image (BLUE CHANNEL).jpg')

fig98 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted AHVD Image (BLUE CHANNEL)');
text(0.5, 0.5, num2str(PSNR_B_OrgAhvd)); %Figure 98: Peak To
Noise Ratio (PSNR) Between Original Image - Encrypted AHVD
Image (BLUE CHANNEL)
saveas(fig98,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted AHVD Image (BLUE CHANNEL).jpg')


fig99 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted AHVD Image (GRAY SCALE)');
text(0.5, 0.5, num2str(mse_OrgAhvd)); %Figure 99: Mean
Square Error (MSE) Between Original Image - Encrypted AHVD
Image (GRAY SCALE)
saveas(fig99,'Mean Square Error (MSE) Between Original Image
- Encrypted AHVD Image (GRAY SCALE).jpg')

fig100 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted AHVD Image (GRAY SCALE)');
text(0.5, 0.5, num2str(PSNR_OrgAhvd)); %Figure 100: Peak To
Noise Ratio (PSNR) Between Original Image - Encrypted AHVD
Image (GRAY SCALE)
saveas(fig100,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted AHVD Image (GRAY SCALE).jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

fig101 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted Original Image (RED CHANNEL)');
text(0.5, 0.5, num2str(mse_R_OrgScrOrg)); %Figure 101: Mean
Square Error (MSE) Between Original Image - Encrypted
Original Image (RED CHANNEL)
saveas(fig101,'Mean Square Error (MSE) Between Original
Image - Encrypted Original Image (RED CHANNEL).jpg')
```

```
fig102 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted Original Image (RED CHANNEL)');
text(0.5, 0.5, num2str(PSNR_R_OrgScrOrg)); %Figure 102: Peak
To Noise Ratio (PSNR) Between Original Image - Encrypted
Original Image (RED CHANNEL)
saveas(fig102,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted Original Image (RED CHANNEL).jpg')


fig103 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted Original Image (GREEN CHANNEL)');
text(0.5, 0.5, num2str(mse_G_OrgScrOrg)); %Figure 103: Mean
Square Error (MSE) Between Original Image - Encrypted
Original Image (GREEN CHANNEL)
saveas(fig103,'Mean Square Error (MSE) Between Original
Image - Encrypted Original Image (GREEN CHANNEL).jpg')

fig104 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted Original Image (GREEN CHANNEL)');
text(0.5, 0.5, num2str(PSNR_G_OrgScrOrg)); %Figure 104: Peak
To Noise Ratio (PSNR) Between Original Image - Encrypted
Original Image (GREEN CHANNEL)
saveas(fig104,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted Original Image (GREEN CHANNEL).jpg')


fig105 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted Original Image (BLUE CHANNEL)');
text(0.5, 0.5, num2str(mse_B_OrgScrOrg)); %Figure 105: Mean
Square Error (MSE) Between Original Image - Encrypted
Original Image (BLUE CHANNEL)
saveas(fig105,'Mean Square Error (MSE) Between Original
Image - Encrypted Original Image (BLUE CHANNEL).jpg')

fig106 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted Original Image (BLUE CHANNEL)');
text(0.5, 0.5, num2str(PSNR_B_OrgScrOrg)); %Figure 106: Peak
To Noise Ratio (PSNR) Between Original Image - Encrypted
Original Image (BLUE CHANNEL)
saveas(fig106,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted Original Image (BLUE CHANNEL).jpg')


fig107 = figure('Name','Mean Square Error (MSE) Between
Original Image - Encrypted Original Image (GRAY SCALE)');
text(0.5, 0.5, num2str(mse_OrgScrOrg)); %Figure 107: Mean
Square Error (MSE) Between Original Image - Encrypted
Original Image (GRAY SCALE)
saveas(fig107,'Mean Square Error (MSE) Between Original
Image - Encrypted Original Image (GRAY SCALE).jpg')
```

```
fig108 = figure('Name','Peak To Noise Ratio (PSNR) Between
Original Image - Encrypted Original Image (GRAY SCALE)');
text(0.5, 0.5, num2str(PSNR_OrgScrOrg)); %Figure 108: Peak
To Noise Ratio (PSNR) Between Original Image - Encrypted
Original Image (GRAY SCALE)
saveas(fig108,'Peak To Noise Ratio (PSNR) Between Original
Image - Encrypted Original Image (GRAY SCALE).jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

fig109 = figure('Name','Entropy Of Original Image');
text(0.5, 0.5, num2str(entropy(imgOrg))); %Figure 109:
Entropy Of Original Image
saveas(fig109,'Entropy Of Original Image.jpg')

fig110 = figure('Name','Entropy Of Encrypted Original
Image'); text(0.5, 0.5,
num2str(entropy(scrambledImageOrg))); %Figure 110: Entropy
Of Encrypted Original Image
saveas(fig110,'Entropy Of Encrypted Original Image.jpg')

fig111 = figure('Name','Entropy Of Encrypted AHVD Image');
text(0.5, 0.5, num2str(entropy(scrambledImageAHVD)));
%Figure 111: Entropy Of Encrypted AHVD Image
saveas(fig111,'Entropy Of Encrypted AHVD Image.jpg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
```