
GAZİ ÜNİVERSİTESİ
Gazi
1926

K5: ÇOK BİÇİMLİ RESİM DESTEKLEYEN BİR STEGANOGRAFİ UYGULAMASI

Ali İlker KOCATEPE

YÜKSEK LİSANS TEZİ
BİLGİSAYAR BİLİMLERİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

ARALIK 2016

ALİ İLKER KOCATEPE tarafından hazırlanan "K5: ÇOK BİÇİMLİ RESİM DESTEKLEYEN BİR STEGANOGRAFİ UYGULAMASI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ / OY ÇOKLUĞU ile Gazi Üniversitesi Bilgisayar Bilimleri Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Hacer KARACAN

Bilgisayar Müh. Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum/~~onaylamıyorum~~ .....

Başkan: Prof. Dr. Şeref SAĞIROĞLU

Bilgisayar Müh. Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum/~~onaylamıyorum~~ .....

Üye: Doç Dr. Osman ABUL

Bilgisayar Müh. Anabilim Dalı, TOBB Ekonomi ve Teknoloji Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum/~~onaylamıyorum~~ .....

Tez Savunma Tarihi: 14/12/2016

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

.....

Doç. Dr. Bünyamin CİYLAN

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Ali İlker KOCATEPE

14/12/2016

K5: ÇOK BİÇİMLİ RESİM DESTEKLEYEN BİR STEGANOGRAFİ UYGULAMASI

(Yüksek Lisans Tezi)

Ali İlker KOCATEPE

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Aralık 2016

ÖZET

Steganografi, iletilmek istenen mesajı sıradan bir taşıyıcıya gizleyerek dikkat çekmeyecek bir yapı oluşturma tekniklerini konu edinir. İletişim teknolojilerindeki gelişmeler sayesinde kişiler arası dosya paylaşımı her geçen gün kolaylaşmakta ve yaygınlaşmaktadır. Resim dosyalarının, iletişim kanalının güvenliği gözetilmeksizin yoğun bir şekilde paylaşılması, bu ortamın steganografik amaçlı kullanımını popüler kılmaktadır. Dönüşüm tabanlı resim steganografisi teknikleri arasında yaygın kullanıma sahip olan F5 algoritması, sıralamalı ayrım ve matris kodlama tekniklerinden faydalanmakla birlikte yalnızca JPEG biçimli stego-resimler üretebilmektedir. Olası bir steganaliz saldırısına karşı taşıyıcı resmin her bölümünün mümkün olduğunca rastgele kullanılması ve veri gömme yoğunluğunun taşıyıcının her yerinde yakın değerde olması, sıralamalı ayrım tekniği ile sağlanır. Matris kodlama tekniği sayesinde, mesaj bitlerini taşıyıcıya gömme esnasında daha az sayıda değişiklik yapılabilmektedir. Bu tez çalışmasının amacı, sıralamalı ayrım ve matris kodlama işlemlerini içeren ve aynı zamanda uzamsal tabanlı yöntemler kullanılması sayesinde çok biçimli resim desteği sağlayan bir steganografi uygulaması geliştirmektir. Geliştirilen K5 Steganografi uygulaması ile veri gizlenen BMP, PNG ve JPEG dosyalarının ki-kare testi ile steganalizi sayesinde, LSB algoritmasından daha iyi ve F5 algoritması ile çok yakın performans sonuçları elde edilmiştir. Böylece K5'in, ki-kare saldırısına karşı fark edilemezliği yüksek ve çok biçimli resim desteği sağlayan bir steganografi uygulaması olduğu ortaya konmuştur.

Bilim Kodu : 902.1.179
Anahtar Kelimeler : Veri gizleme, resim steganografisi, sıralamalı ayrım, matris kodlama
Sayfa Adedi : 85
Danışman : Doç. Dr. Hacer KARACAN

K5: A STEGANOGRAPHY APPLICATION WITH MULTI-FORMAT IMAGE SUPPORT

(M. Sc. Thesis)

Ali İlker KOCATEPE

GAZİ UNIVERSITY

INFORMATICS INSTITUTE

December 2016

ABSTRACT

Steganography deals with the techniques which create an unobtrusive structure by hiding the message to be transmitted into an ordinary cover. Due to the developments in communication technologies, file sharing between individuals becomes easier and more common by the day. The fact that image files are shared intensively without consideration of whether the transmission channel is safe or not have made the image medium a popular one for the use of steganographic means. With widespread usage among transformation domain image steganography methods, F5 algorithm utilizes permutative straddling and matrix encoding techniques. However, F5 produces stego-images only in JPEG format. Permutative straddling provides the most possible random use of every part in the cover image when embedding and ensures that the data embedding intensity has similar values throughout the cover image against a possible steganalysis attack. Matrix encoding enables to make less changes when message bits are embedded into the cover. The purpose of this thesis study is to develop a steganography application which includes permutative straddling and matrix encoding features as well as using spatial domain methods to provide multi-format image support. The performance results obtained by applying chi-square steganalysis test on BMP, PNG and JPEG files in which data is hidden via the developed K5 Steganography application are better than LSB algorithm's resulsts and are very close to F5 algorithm's resulsts. Thus, it is stated that K5 is a steganography application which is highly imperceptible against the chi-square attack and has multi-format image support.

Science Code : 902.1.179
Key Words : Data hiding, image steganography, permutative straddling, matrix encoding
Page Number : 85
Supervisor : Assoc. Prof. Dr. Hacer KARACAN

TEŞEKKÜR

Çalışmam boyunca yardımlarını esirgemeyen danışmanım Doç. Dr. Hacer KARACAN'a, birlikte çalıştığımız süre boyunca destekleri ile beni cesaretlendiren tüm Araştırma Görevlisi arkadaşlarıma ve bu çalışmanın tamamlanmasına katkıda bulunan Gazi Üniversitesi Bilişim Enstitüsü yönetimi ile idari personeline en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

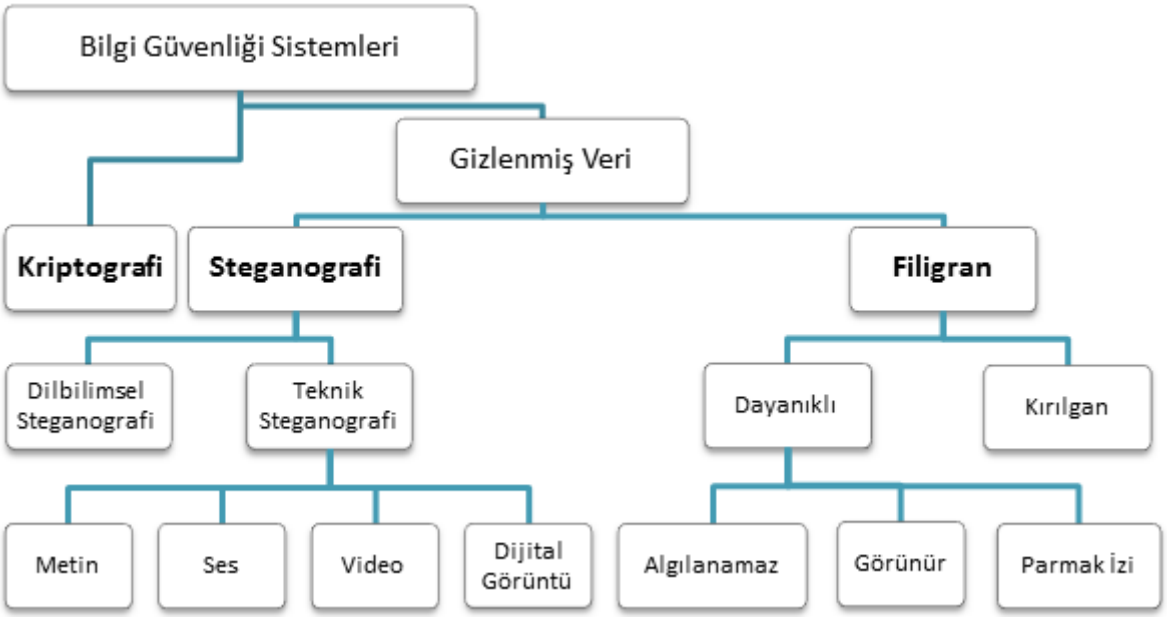
Simgeler	**Açıklama**
bpc	değişiklik başına bit (bits per change)
C#	C Sharp programlama dili

Kısaltmalar	**Açıklama**
ASCII	Bilgi Değişimi için Amerikan Standart Kodu (American Standard Code for Information Interchange)
BMP	Bit Eşlem Resmi (Bitmap Image)
DCT	Ayrık Kosinüs Dönüşümü (Discrete Cosine Transform)
HAS	İnsan İşitme Sistemi (Human Auditory System)
JPEG	Birleşik Fotoğraf Uzmanları Grubu (Joint Photographic Experts Group)
LFSR	Doğrusal Geri-beslemeli Kayan Kaydedici (Linear-Feedback Shift Register)
LSB	En Önemsiz Bit (Least Significant Bit)
MSB	En Önemli Bit (Most Significant Bit)
PNG	Taşınabilir Ağ Grafikleri (Portable Network Graphics)
XOR	Özel Veya İşlemi (Exclusive Or Operation)

1. GİRİŞ

Bilginin, gönderici ve alıcı haricinde kimsenin fark edemeyeceği şekilde iletilebilmesi için tarih boyunca çeşitli yöntemler uygulanmıştır. En sade ifade ile veri gizleme bilimi olarak tabir edilen steganografi, iletilmek istenen mesajı sıradan bir taşıyıcıya gizleyerek dikkat çekmeyecek bir yapı oluşturma tekniklerini konu edinir. Güvenlik gerektiren verilerin güvensiz ortamda iletilmesi işlemi, günümüzde de üstesinden gelinmeye çalışılan bir problemdir.

Korunması gerekli görülen bilginin güvenliğini sağlamak için geliştirilmiş olan sistemler üç ana kategori altında incelenebilir. Şifreleme (kriptografi), filigran (watermarking) ve steganografi olarak sıralanabilecek bu sistemler kullanım amaçları ve uygulanma tekniklerinde farklılaşmaktadır. Kriptografi, steganografi ve filigran arasındaki ilişki Şekil 1.1'de görülmektedir [1].



Şekil 1.1. Bilgi gizlemenin farklı uygulama disiplinleri [1]

Steganografi ile bazı yakın kavramlara sahip olan ve birbiri ile benzer yöntemleri kullanan filigran ve kriptografi teknikleri hakkında ön bilgi sahibi olmak, ilerideki bölümlerde detaylandırılacak konular için faydalı görülmektedir.

Filigran (Watermarking)

Basılı bir ürüne, üreticisini belli etmek amacıyla eklenen özel tasarımlı sembole filigran denir. Sayısal ortamda ise metin, ses veya görüntü dosyalarına çeşitli uygulamalar yoluyla filigran eklenebilmektedir. Duyu organlarınca algılanabilir filigranlar olduğu gibi, yalnızca bilgisayar yazılım veya donanımları tarafından okunabilen filigranlar da mevcuttur. Dijital filigranlar; telif hakkı bilgisi taşıma, kimlik doğrulaması, takip, gözlem ve güvenlik artışı amacıyla kullanılmaktadır [2].

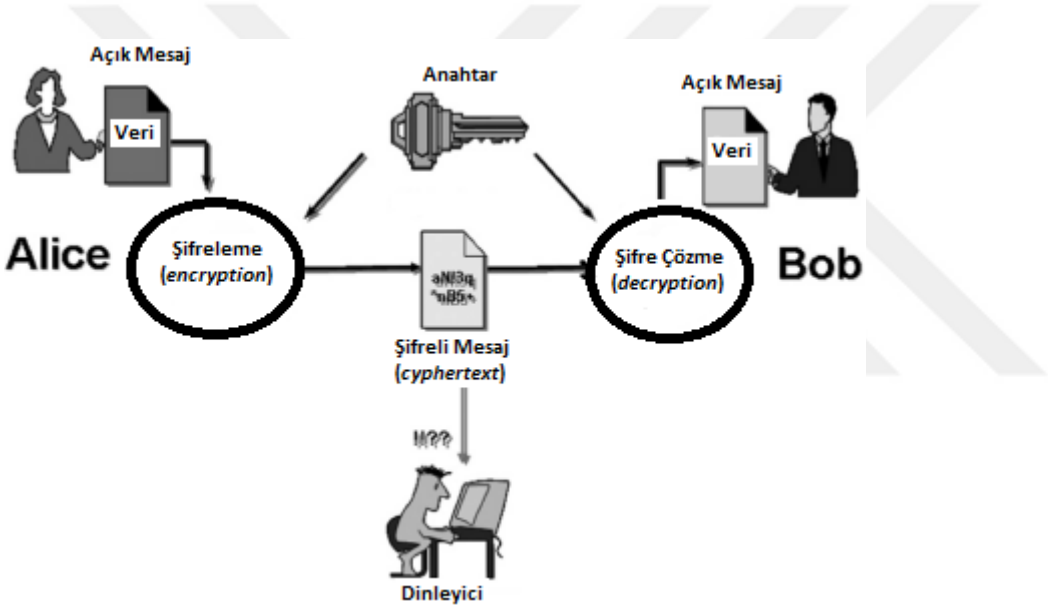
Filigran yöntemi ile steganografi arasındaki en önemli fark, filigranlanan nesnenin taşıdığı özelliği bariz bir şekilde belli etmesine karşın steganografik teknikler uygulanmış nesnenin yetkisiz kişilerce fark edilemezlik amacına sahip olmasıdır [3]. Steganografi ve filigran tekniklerinin karşılaştırması Çizelge 1.1'de görülmektedir [4].

Çizelge 1.1. Filigran ve steganografinin özelliklerinin karşılaştırması [4]

	Filigran	Steganografi
Koruma	Taşıyıcıyı korur	Gizli bilgiyi (yükü) korur
Gizlilik	Gereksinimlere göre; görünmezlik veya algıya bağlı görünürlük	Gömülü bilgi, istenmeyen kişilere karşı görünmezlik özelliği taşır
Dayanıklılık tipi	Değiştirme veya silmeye karşı dayanıklılık	Tespite karşı dayanıklılık
Sinyal işleme / rastsal hata / sıkıştırma etkileri	Filigran kaybına yol açmamalıdır	Gizli bilgi kaybına yol açabilir
Taşıyıcı nesne tipi	Dijital dosyalar – metin, ses, görüntü veya video	Dijital verinin kullanıldığı herhangi bir hizmet, protokol, dosya veya ortam

Kriptografi

Veriyi, ters işlemi mümkün olan bir metot kullanarak, anlaşılmaz bir hale dönüştüren bilgi güvenliği tekniğine kriptografi denir [5]. Kriptografide; şifrelenecek veriye *açık mesaj* (plaintext), veriyi anlaşılmaz hale dönüştürme işlemine *şifreleme* (encryption), şifreleme sonucu elde edilen veriye *şifreli mesaj* (cyphertext), şifreli mesajı tekrar anlamlı hale getirme işlemine *şifre çözme* (decryption), şifreleme ve şifre çözme işlemlerinde kullanılan koda ise *anahtar* (key) adı verilir. Gönderici ve alıcı haricindeki bir kişinin (*dinleyici*nin), şifrelenmiş veriyi elde edebilmesi için anahtarı ve şifreleme algoritmasını bilmesi gerekir. Şekil 1.2'de kriptografinin şematik gösterimi verilmiştir [6].



Şekil 1.2. Kriptografi şeması [6]

Kriptografi ve steganografi tekniklerinin ikisi de, yetkili alıcılar haricinde mesaj içeriğine erişimin engellenmesini amaçlar. Kriptografi uygulanan mesaj şifrelenmiş olsa da varlığı bilinmesine rağmen steganografi uygulanan mesajın varlığı ve hatta iletim yolu bile gizlenmiş olabilir [4,7,8]. Bu iki gizli iletişim tekniği ayrı ayrı veya bir arada kullanılabilir [9]. Bazı kriptografi uygulamaları, şifreleme sonucunda üretilen verinin tamamen rastsal (gürültü) gibi görünmesinden ötürü, steganografik olarak da sınıflandırılabilir [10]. Steganografi ve kriptografi tekniklerinin karşılaştırması Çizelge 1.2'de görülmektedir [4].

Çizelge 1.2. Kriptografi ve steganografinin özelliklerinin karşılaştırması [4]

	Kriptografi	Steganografi
Amaç	İletişim içeriğini karmaşıklaştırmak	İletişimin varlığını gizlemek
Gizlilik	Şifreli mesaj okunamaz haldedir	Gömülü mesaj görünmezdir
İletişimin güvenliği	Anahtarın gizliliğine bağlıdır	Kullanılan algoritmanın gizliliğine bağlıdır
Dayanıklılık güvencesi	Şifreleme algoritmasının karmaşıklığı	Algısal ve/veya istatistiksel görünmezlik / protokole uyum
Saldırılar	Tespiti kolay, çözümü karmaşık	Tespiti ve çözümü karmaşık
Teknik önlemler	Tersine mühendislik	İletilen verinin sürekli izlenmesi ve analizi
Yasal önlemler	Kriptografi ihracı kanunları	Katı cihaz/protokol belirtimleri

Bu tez çalışması ile amaçlanan, sıralamalı ayrım ve matris kodlama işlemlerini içeren ve aynı zamanda çok biçimli resim desteği sağlayan bir steganografi uygulaması geliştirmektir. Önerilen K5 Steganografi uygulaması için ki-kare saldırısına karşı fark edilemezliği yüksek olması sebebiyle F5 algoritması model alınmış ve bu algoritmanın yalnızca JPEG biçimli stego-resimler üretebilmesi sınırlılığını ortadan kaldırmak hedeflenmiştir.

Çalışmanın ikinci bölümünde öncelikle steganografi hakkında detaylı bilgi verilerek, steganografik metodolojinin temeli sayılabilecek olan Tutsakların Problemi (The Prisoners' Problem) açıklanmıştır. Ardından, steganografinin tarihçesine değinilerek geçmişten günümüze steganografi örnekleri verilmiştir. Daha sonra steganografide kullanılan tanımlar açıklanmış ve kullanılan ortama göre steganografi türleri, alt türleri de kapsayacak şekilde sıralanmıştır. Son olarak resim steganografisinde yaygın kullanılan

algoritmalar olan LSB ve F5 hakkında ayrıntılı bilgi verilmiştir. Geliştirilen K5 Steganografi uygulamasının model aldığı F5 algoritmasında kullanılan sıralamalı ayrım ve matris kodlama özellikleri detaylı açıklanmıştır.

Üçüncü bölümde, steganografik sistemi kırma işlemi olan steganaliz ele alınmıştır. Steganaliz yöntemlerinin temel mantığı ortaya konarak steganaliz yaklaşımları hakkında bilgi verilmiştir. Bazı steganaliz örnekleri sunularak devam edilmiş ve steganalizde kullanılan kategorik saldırı türleri açıklanmıştır. Ayrıca istatistiksel saldırılardan ki-kare testi detaylı anlatılmıştır.

Önerilen K5 Steganografi uygulaması hakkında ön bilgiler verilerek başlanan dördüncü bölümde, çalışmanın yöntemi ve çalışmada kullanılan araçlar belirtilmiştir. Bölümün devamında ise K5 uygulamasının girdileri ve çıktıları, veri gömme (embedding) ve veri ayıklama (extraction) aşamalarındaki tüm işlemler detaylı olarak anlatılmıştır.

Çalışmanın beşinci bölümünde, K5 Steganografi uygulaması ile LSB ve F5 algoritmaları kullanan uygulamaların karşılaştırılması amacıyla ki-kare saldırısı içeren üç farklı steganaliz testi gerçekleştirilmiştir. Saldırı için geliştirilen steganaliz uygulaması anlatılarak, testler sonucunda elde edilen bulgular karşılaştırmalı görseller ile sunulmuştur.

Son bölümde ise bir önceki bölümde elde edilen veriler yorumlanarak anlamlandırılmıştır. K5 Steganografi uygulamasının avantajları ve sınırlılıkları ortaya konularak, iyileştirilebilecek yönleri ve ileri çalışmalar önerilmiştir.

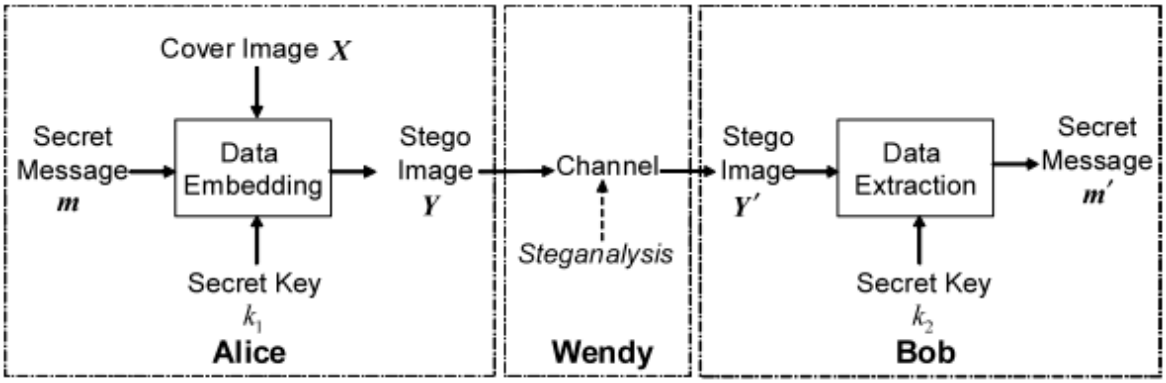
K5 Steganografi ve Ki-kare Steganaliz uygulamalarında kullanılan metotların C# programlama dilindeki kaynak kodları, Ekler kısmında sunulmuştur.

2. STEGANOGRAFİ

Steganografi kelimesi, Yunanca'da "gizli" anlamına gelen steganos (στεγανός) ile "yazılan, çizilen" anlamındaki graphein (γράφειν) sözcüklerinin birleşiminden türemiştir [9]. Bir nesne içerisine, dışarıdan bir gözlemci için görünür olmayacak şekilde mesaj gizleme işlemleri için steganografi terimi kullanılmaktadır [11]. Bir başka ifade ile steganografi, taşıyıcı sinyalde herhangi bir algısal değişikliğe neden olmadan mesaj sinyalini gizleme olarak tanımlanabilir [12].

Kelime olarak steganografinin ilk kullanımı 15. yüzyılda yaşamış olan Johannes Trithemius'un yazdığı üç ciltlik kitapta (Steganographia: hoe est ars per occultam scripturam animi sui voluntatem absentibus aperiendi certa) bulunmaktadır [11]. Astroloji ile ilgili olduğu düşünülen üçüncü ciltte, sayılar içeren tablolar bulunmaktadır. Yapılan inceleme sonucunda bu tablolarda gizlenmiş mesajlar olduğu açığa çıkarılmıştır.

Her iletişim türü bir kanala veya iletim ortamına ihtiyaç duyar. Steganografik iletişimin de gizlenmiş mesajın iletimi için bir kanal gereksinimi vardır. Simmons, 1984'te yaptığı çalışmasında, alıcı ve verici arasında gizlenmiş veri iletimi için kullanılabilecek kanal oluşturulabileceğini ispatlamıştır [13]. Tutsakların Problemi (The Prisoners' Problem) olarak bilinen bu çalışma, temel steganografik model kabul edilir. Bu problemi temsil eden steganografi modelinin blok şeması Şekil 2.1'de gösterilmiştir [8].



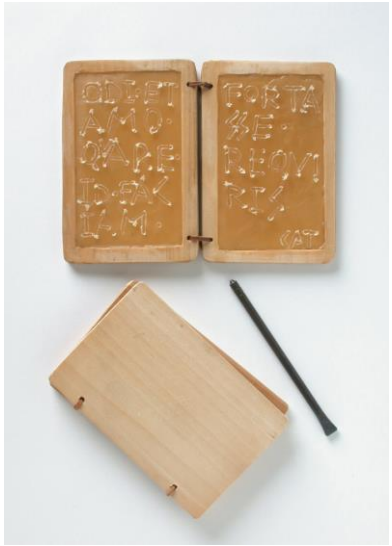
Şekil 2.1. Tutsakların Problemi ve steganografi modeli [8]

Bu modele göre; Alice ve Bob, hapishaneden kaçış planı yapması gereken iki tutsaktır ve aralarındaki iletişim Wendy isimli gardiyan tarafından izlenmektedir. İletilmek istenen

gizli mesaj (m), k_1 anahtarı kullanılarak taşıyıcı nesneye (X) gömülmekte ve sonuçta stego-nesne (Y) elde edilmektedir. Wendy'nin izlediği ve iletişime müdahalede bulunabileceği kanal ile stego-nesne (Y'), Bob'a ulaşır. Burada eğer $Y' = Y$ ise müdahale olmamıştır ve Wendy pasif gardiyan konumundadır. Eğer $Y' \neq Y$ ise bir müdahale söz konusudur ve Wendy'nin aktif gardiyan durumunda olduğu anlaşılır. Alıcı (Bob), k_2 anahtarını ve stego-nesneyi (Y') kullanarak gönderilen gizli mesajı (m') elde etmeye çalışır. $m = m'$ olduğu durumda, steganografik sistem doğru çalışıyor demektir.

Steganografinin örneklerini antik çağlarda dahi görmek mümkündür. Yunan tarihçi Herodot, avlanan tavşan cesetlerinin içine gizlenmiş mesajlardan bahsetmektedir. MÖ 440'da ise Milet hükümdarı Histiæus, Pers kralından gizli olarak göndermek istediği mesajı, saçlarını kazıttığı kölesinin başına dövme ile yazdırmıştır. Saçları uzadıktan sonra mesajı taşıyan kişi şüphe çekmeden alıcıya ulaşmıştır.

Aynı yüzyılda Heroclotus tarafından nakledilen anekdota göre Demaratus, Xerxes'in istilasına karşı uyarmak istediği Spartalı alıcılara göndermek istediği mesajı, o yıllarda yazım için kullanılan tahta kaplı mum tabletler vasıtasıyla iletmiştir. Bu yöntemde mum kazınarak, iletilmek istenen mesaj tahta üzerine yazılmış ve tekrar mum ile kaplanmıştır. Temsili mum tablet Resim 2.1'de görülmektedir.



Resim 2.1. Temsili mum tablet

Parşömenin sık kullanıldığı dönemlerde ise steganografi aracı olarak görünmez mürekkep ile yazma örneklerine rastlanmaktadır. Örneğin, bir kaktüs türü olan tithymalus bitkisinin özünden elde edilen mürekkep kuruduğunda gözle görünmez hale gelmektedir.

Sık rastlanan steganografi tekniklerinden biri de, mucidi olan İtalyan matematikçi Girolamo Cardano'nun adı ile anılan Cardano Izgarası tekniğidir. Üzerinde rastgele gibi gözüken ancak planlı açılmış delikler bulunan ızgara şeklindeki kağıt veya metal levha, sıradan bir metin yazılı olan kağıdın üzerine yerleştirildiğinde, deliklerden görünen harfler gizlenmiş mesajı ortaya çıkarır.

Daha sonraları ise mesajı gizlemek için yazılan metnin kendisinin kullanımı popüler hale gelmiş ve böylece metin steganografisinin ilk örnekleri ortaya çıkmıştır. Aldus Manutius tarafından yazılan Hypnerotomachia Poliphili isimli kitapta, ilk 38 bölümün baş harfleri birleştirildiğinde "Peder Francesco Colonna, Polia'ya tutku ile aşıktır" mesajına ulaşılmaktadır [14].

Rönesans dönemine gelindiğinde ise İtalyan bilim adamı Giambattista della Porta tarafından uygulanan bir steganografi tekniği göze çarpmaktadır. Şap ve sirke karışımı ile oluşturulan mürekkep kullanılarak haşlanmış yumurtanın kabuğu üzerine gizli mesaj yazıldığında, kabuk yüzeyinde herhangi bir iz kalmamasına rağmen yumurtanın beyazı üzerinde renk değişimi meydana gelmektedir. Alıcı, şüphe çekmeyecek şekilde kendisine ulaştırılan yumurtanın kabuğunu soyarak mesajı okuyabilmektedir.

1680 yılında Gaspar Schott, yayınladığı Schola Steganographica adlı kitabında, müzik notalarını kullanarak mesaj gizleme yöntemini anlatmıştır. Her bir notanın bir harfe karşılık olarak kullanıldığı bu yöntem ile dikkat çekmeden veri iletimi yapılabilmesine karşın, oluşturulan melodi çalınmak istediğinde olağan dışı bir durum olduğu ortaya çıkabilecektir [4].

II. Dünya Savaşı ile birlikte steganografik yöntemler gelişim göstermiştir. Bu tarihlerdeki günlük bir gazeteden alıntı aşağıdaki metin, balıkçılık ile ilgili bir paragraftır.

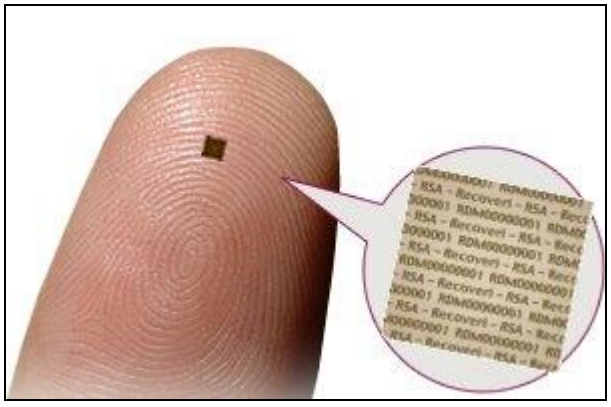
Fishing freshwater bends and saltwater coasts rewards anyone feeling stressed. Resourceful anglers usually find masterful leapers fun and admit swordfish rank overwhelming anyday.

Ancak metindeki kelimelerin üçüncü harfleri sıralandığında, casusluk amacıyla iletilmek istenen aşağıdaki mesaj elde edilmektedir:

SEND LAWYERS GUNS AND MONEY

1942 yılında Velvalee Dickinson isimli ABD vatandaşı kadın, sahibi olduğu oyuncak bebek dükkanı sayesinde, oyuncak bebekler ile ilgili dikkat çekmeyecek mektuplar yazarak bunları çeşitli adreslerden göndermiştir. Mektup metni ve gönderi adresleri sayesinde ABD donanması hakkında bilgileri Japonya'ya sızdırdığı ortaya çıkıp casusluk ile suçlandığında ise *Doll Woman* lakabını almıştır.

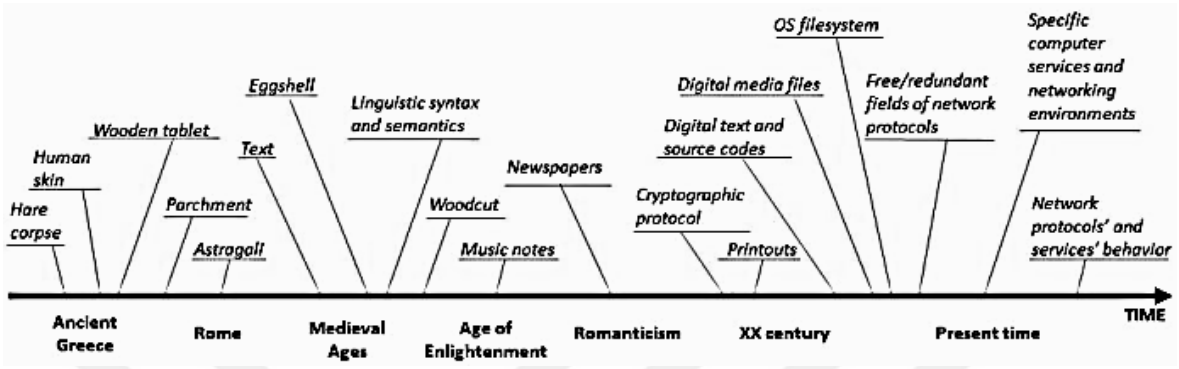
Yine II. Dünya Savaşı sırasında Nazi casuslarınca sık kullanılan steganografi yöntemlerinden biri de mikronokta (microdot) tekniğidir. Mikrofilm yapısında 1/200 ölçeğinde küçültülmüş fotoğraf kareleri, daktilo ile yazılmış metinlere nokta (.) olarak eklenerek gizli iletişimde kullanılmıştır. Resim 2.2'de örnek bir mikronokta görülmektedir.



Resim 2.2. Mikronokta örneği

1966 yılında Kuzey Vietnam tarafından esir tutulan Amerikalı asker Jeremiah Denton, televizyondan yayınlanan bir basın konferansı sırasında kendisine zorlanan metni okurken gözlerini belirli aralıklarla kırparak, Mors kodu ile *TORTURE* (işkence) mesajını iletmiştir.

İletişim ortam ve araçlarının tarihsel gelişimine paralel bir ilerleme kaydeden steganografi, dijital dünyaya adapte olarak günümüzde de kullanılır hale gelmiştir. Steganografi kullanımının zamana bağlı gelişimi Şekil 2.2'de gösterilmiştir [4].



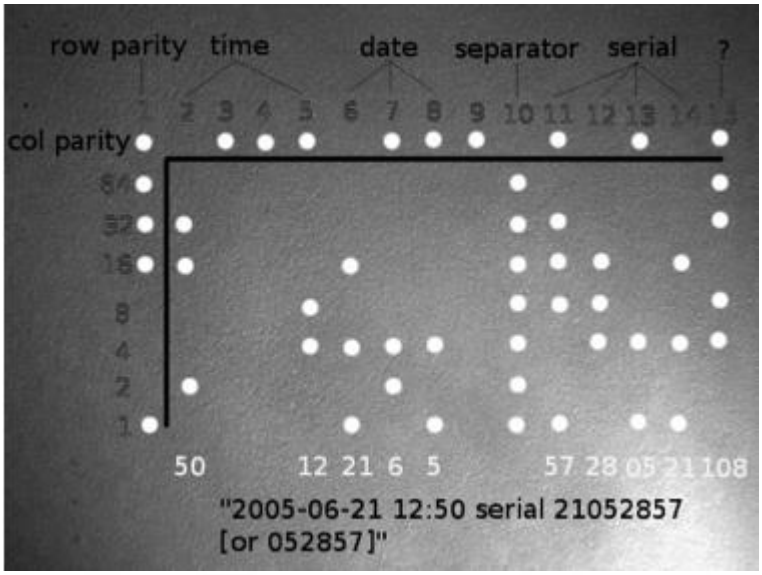
Şekil 2.2. Steganografi gelişiminin zaman çizelgesi [4]

Metin üzerinde steganografi uygulamasına bir örnek olarak, dijital ortamda yazılan kelimeler arasında bırakılan fazladan boşluklar sayesinde gizli veri iletimi verilebilir. Bu teknik sayesinde, İngiltere'de Margaret Thatcher zamanında kabine belgelerinin sızdırılmasının izlendiği iddia edilmektedir [15].

Özellikle 11 Eylül 2001 saldırılarından sonra, El Kaide tarafından steganografi kullanılarak İnternet kanalı ile gizli mesaj iletiminin yapılabilirliği konusu gündeme gelmiştir. Michigan Üniversitesi'nden bir grup araştırmacı bu konu üzerine çeşitli web sitelerinde bulunan iki milyondan fazla görüntü dosyasını incelemiş ancak şüpheli bir duruma rastlamadıklarını raporlandırmışlardır [16].

Sağıroğlu ve Tunçkanat, 2002'de yaptıkları çalışma ile İnternet üzerinden güvenli iletişime imkan tanıyan TurkSteg uygulamasını geliştirmişlerdir [17].

2004 yılında PCWorld tarafından yayınlanan bir makale, DocuColor (Xerox) model yazıcılar kullanılarak yazdırılan her sayfada açık sarı renkte gizli noktalar olduğunu ortaya çıkarmıştır [18]. Gözle fark edilemeyen bu noktalar, mavi ışık ve büyüteç kullanılarak görülebilmektedir. Electronic Frontier Foundation, bu noktalar ile yazıcının seri numarası, belgenin yazdırıldığı tarih ve saat bilgilerinin kodlandığını ortaya çıkarmıştır. Görülebilir hale getirilmiş sarı noktalı kağıt örneği Resim 2.3'de verilmiştir.



Resim 2.3. Sarı noktalar ile izleme yöntemi kullanan yazıcının çıktı örneği [18]

Kelime işlemci yazılımları yoluyla uygulanan bir başka steganografi tekniğinde ise, metinde önceden belirlenmiş bazı harflerin renk değerleri 1 birim arttırılarak veri gizlenmesi sağlanmaktadır [19].

2006 yılında Şahin, Buluş ve Sakallı tarafından yapılan çalışmada gri seviye resimler üzerinde veri gizlemek için LSB'lerin sıralı seçimi yerine, sayı teorisi ve ayrık logaritma fonksiyonu sayesinde rastgele seçim yapılması sağlanmıştır [20].

Ulutaş, Nabiyev ve Ulutaş, 2009'da geliştirdikleri yöntem sayesinde Blakley'in 1979'da tasarladığı geometrik tabanlı gizli paylaşım yaklaşımını steganografi ile iyileştirerek resim steganografisinde kapasite artışı sağlamışlardır [21].

Ünlü, 2012'de yaptığı çalışma sonucunda ortamdan ve yöntemden bağımsız bir steganografik kütüphane tasarımı ortaya koymuştur [22]. Bu tasarıma dayanarak Ünlü ve Karacan tarafından 2013'te geliştirilen kafes yaklaşımı sayesinde ise, gerçek zamanlı video dosyalarının resim ve ses alanları birlikte kullanılarak karmaşıklık ve güvenlik seviyeleri arttırılmış bir steganografik model ortaya konmuştur [23].

Bansal ve Chhikara, 2014'de önerdikleri Shield (Kalkan) Algoritması yöntemiyle, DCT tabanlı steganografide mutlak değeri sıfır olmayan tüm katsayılara veri gizlemek yerine katsayıların veri kapasitelerini dikkate alarak gizleme yapmayı öne sürmüşlerdir [24].

2015'deki çalışması ile Karakış, MRG ve EEG görüntülerinin paylaşılması sırasında hasta bilgilerinin mahremiyetini korumak amacıyla, bu bilgilerin bulanık mantık tabanlı bir steganografi tekniğiyle paylaşılan görüntüler içerisine gizlenmesine olanak sunan bir uygulama geliştirmiştir [25].

Durdu ve Özcerit, 2015 yılında yayınladıkları çalışmada, taşıyıcı resmi 4 baytlık çerçevelere bölerek eşleştirme alanları oluşturdukları ve LSB değişimi yerine LSB eşleştirme esasına dayalı bir steganografik sistem geliştirmişlerdir [26].

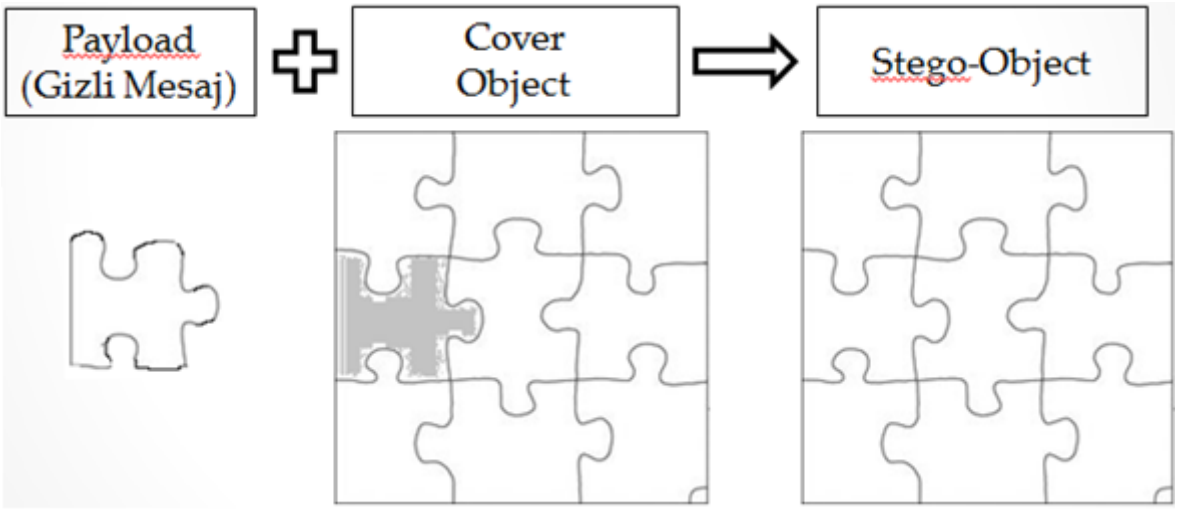
Verilen örnek olay ve çalışmalar haricinde dijital steganografi birçok farklı uygulamada da kullanılmaktadır. Telif hakkı koruması, görüntü arama motorlarının iyileştirilmesi ve videolarda görüntü-ses senkronizasyonu bu uygulamalara örnektir [1]. Ayrıca bir fotoğrafa başlık, tarih, yer ve fotoğraftaki kişilerin isimleri gibi bilgilerin eklenmesi de steganografi ile mümkün olabilmektedir [27].

2.1. Temel Kavramlar

Uygulama ortam ve araçları ile kullanılan tekniklere göre kimi zaman farklı şekillerde ifade edilseler de, steganografi alanında etkili çalışma için bazı tanımlamalara ihtiyaç duyulur. Ayrıca steganografik teknikler kullanılarak oluşturulan bir mesajın, yalnızca belirlenmiş alıcılar tarafından elde edilebilir olması için bazı özellikleri barındırması gerekir.

2.1.1. Taşıyıcı nesne, yük ve stego-nesne

Gizlenecek veriyi taşıyacak olan metin, ses veya imaj (resim veya video) dosyası *taşıyıcı nesne* (cover object) olarak adlandırılır. Taşıyıcı nesne içine gizlenecek veri bitleri *yük* (payload) olarak ifade edilir. Yük içeren taşıyıcı nesne, yani veri bitlerinin gömülmüş olduğu dosya artık *stego-nesne* (stego-object) olarak isimlendirilir. Taşıyıcı nesne, yük ve stego-nesne arasındaki ilişki Şekil 2.3'de gösterilmiştir.



Şekil 2.3. Taşıyıcı nesne, yük ve stego-nesne

Yükün taşıyıcı nesneye dahil edilmesi işleminde taşıyıcı nesnenin istatistiksel özellikleri korunarak, stego-nesne ile arasındaki fark en aza indirgenmeye çalışılır [7].

2.1.2. Kapasite

Taşıyıcı nesnenin bütünlüğü bozulmadan ne kadar yük içerebileceği kapasite (capacity) deyimiyle ifade edilir [7]. Kapasite, yükün mutlak miktarı (byte) olarak anlaşılabileceği gibi oransal bir değer (örn. bit/piksel) de belirtebilir [8].

2.1.3. Fark edilemezlik

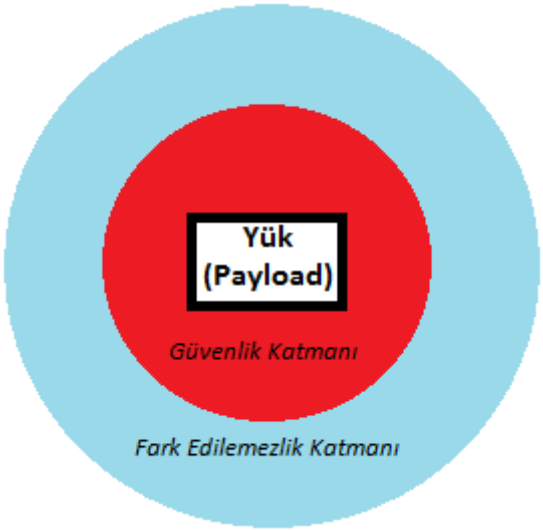
Fark edilemezlik (imperceptibility), stego-nesnenin yetkili olmayan kişilerce dikkat çekmemesi gerektiğini belirten özelliktir. İyi bir steganografi uygulaması, gizli bir verinin var olduğu şüphesini uyandırmamalıdır [8].

Stego-nesneye yapılması planlanan bir karşı saldırı (steganaliz) durumunda saldırganın aşması gereken ilk katman fark edilemezlik olacaktır.

2.1.4. Güvenlik

Fark edilemezlik katmanının aşılması halinde, gizlenmiş verinin açığa çıkarılmasının zorluğu güvenlik (security) ile ilgilidir. Stego-nesnede güvenlik özelliği çoğunlukla kriptografi teknikleri ile sağlanır [8].

Bir steganografik sistemdeki fark edilemezlik ve güvenlik katmanları, Şekil 2.4'de şematize edilmiştir.



Şekil 2.4. Steganografik sistemde fark edilemezlik ve güvenlik

2.1.5. Dayanıklılık

Bu kavram sıklıkla güvenlik ile karıştırılarak kullanılsa da, iki kavramın ifade ettikleri farklıdır. Gizlenmiş mesajın fark edilmesi, ancak güvenlik katmanının aşılamayarak verinin elde edilememesi durumunda saldırganlar, mesajın alıcıya ulaşmaması için içeriğini değiştirmeye veya gizli içeriği tamamen yok etmeye çalışabilirler. Böyle bir durumda stego-nesnenin, içerdiği yükü bozmaya yönelik saldırılara karşı gösterdiği direnç, dayanıklılık (robustness) özelliği ile ifade edilir [7].

2.2. Kullanılan Ortama Göre Steganografi Türleri

Günümüzde dijital steganografinin kullanıldığı ortamlar metin (text), ses (audio) ve durağan (still) ile hareketli (video) olmak üzere görüntü (image) olarak sıralanabilir. Ayrıca ağ (network) ortamında da steganografi uygulamaları geliştirilebilmektedir.

2.2.1. Metin steganografisi

İletişimde basitlik sağladığı ve fazla dikkat çekmeyeceği düşünüldüğü için metin steganografisi halen tercih edilebilmektedir. Metin steganografisinde dikkate alınması gereken temel gereksinimler, okunabilirliğin korunması ve stego-metin içeriğinin normal görünmesidir. Bu steganografi türünün avantaj ve dezavantajları Çizelge 2.1'de gösterilmiştir.

Çizelge 2.1. Metin steganografisinin avantaj ve dezavantajları

Avantajlar	Dezavantajlar
+ Kayıpsız sıkıştırma teknikleri uygulanabilir	- Artık bit bulma zorluğu (metinde gereksiz bilgi azlığı)
+ Daha az dikkat çeker (yüksek fark edilemezlik)	- Düşük kapasite
+ Daha az yer kaplar (düşük dosya boyutu)	
+ Basit iletişim olanakları	

Metin steganografisi, üç alt kategoride incelenebilir.

Yapısal (structural)

Boşluk ( ) veya altçizgi (_) karakterleri eklemek suretiyle metnin fiziksel formunu değiştirmek, yazı tipi (font) boyutunu gözle fark edilemeyecek şekilde değiştirmek, kasıtlı

olarak yanlış yazılmış kelimeler kullanmak vb. işlemler ile yapısal metin steganografisi gerçekleştirilir.

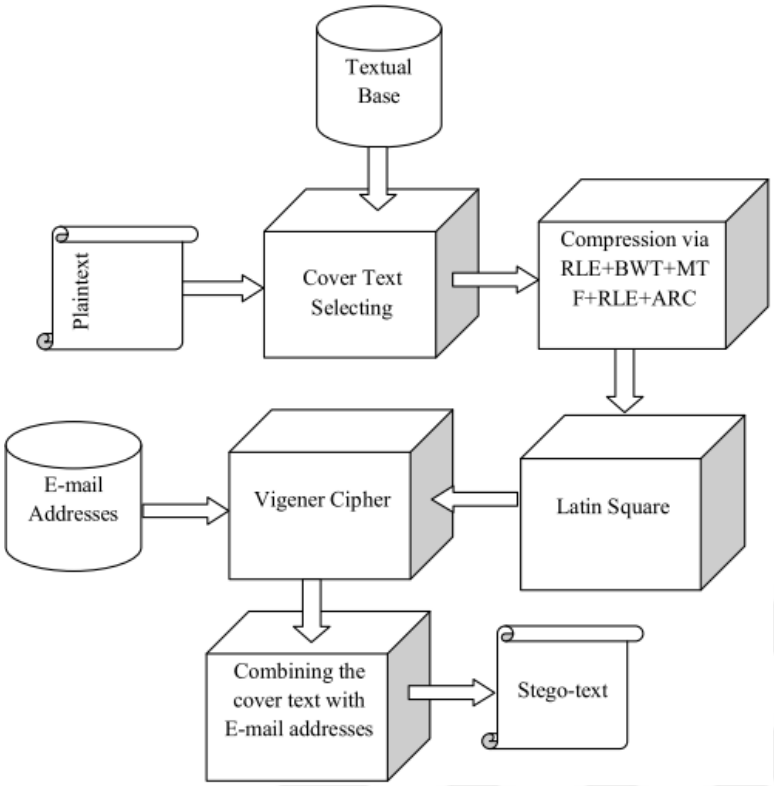
Dilsel (linguistic)

Metin dilinde bulunan noktalama gibi sintaks kurallarını kasıtlı değiştirerek uygulanabileceği gibi; içerikteki kelimelerin eş anlamlıları ile değişimi yoluyla semantik olarak da uygulanabilir.

Rastsal (random) ve istatistik tabanlı

Yükün gömüleceği örtü metni (cover-text) belirlenirken, tamamen rastgele ya da planlı istatistiksel hesaplamalara göre seçim yapılır.

Hassan tarafından 2015 yılında tasarlanan e-posta tabanlı metin steganografisinde; örtü metni olarak, hazırlanan e-posta havuzundan maksimum tekrarlı örnek değişkeni dikkate alınarak bir e-posta seçilir. Gizlenecek mesaj ile seçilen örtü metni arasındaki uzaklık matrisinin hesaplanması sonrası çeşitli sıkıştırma teknikleri uygulanır. Latin Karesi ve Vigenere Şifresi teknikleri kullanılarak yapılan e-posta adresi seçimi, stego-anahtar işlevini görür. Böylece hazırlanan stego-metin hem alıcıya hem de stego-anahtar olan sahte e-posta adresine aynı anda gönderilir. Tasarlanmış olan steganografik sistem, Şekil 2.5'te gösterilmiştir [28].



Şekil 2.5. E-posta tabanlı metin steganografisinin blok şeması [28]

2.2.2. Ses steganografisi

İnsan işitme sistemi (HAS – Human Auditory System) bazı algısal yanılsamalara eğilimli bir yapıdır. Ayrıca bazı frekanstaki sesler insan kulağı tarafından duyulamamaktadır. HAS'ın düşük düzeyli ayırdediciliğinden ötürü, yüksek sesler alçak sesleri maskeleyebilmektedir. Ayrıca HAS, sesler arasındaki faz farkını göreceli olarak ayırdedebilse de sesin mutlak faz değerini algılayamamaktadır [29].

Bu bilgiler üzerine geliştirilen ses ortamındaki steganografi uygulamalarında; frekans maskeleme, yankı (eko) gizleme, faz kodlama, yama ekleme ve spektrum yayma gibi teknikler kullanılır [4].

Değişen ses şiddetine göre gürültüye gizlenecek veri miktarını ayarlamak için adaptif veri sönümlenmesi (adaptive data attenuation), taşıyıcı olarak hata düzeltme kodlaması (ECC - Error Correction Coding) kullanımı ve ses içeriğine bağlı olarak ses sinyalinin ardışık örneklemlerindeki değişim miktarının hesaplanması da ses steganografisi alanında kullanılabilecek tekniklerdendir [29].

2.2.3. Görüntü (resim ve video) steganografisi

Steganografi ortamı olarak görüntü seçildiğinde uygulanabilecek teknikler; uzamsal tabanlı, dönüşüm tabanlı, spektrum genişletme ve model tabanlı olmak üzere dört başlık altında incelenebilir.

Uzamsal tabanlı (spatial domain) teknikler

Bu teknikler genel olarak taşıyıcı resmin piksel değerlerini değiştirerek gizli mesajı taşıyıcı resme gömme esasına dayanır [1]. Bir resmin piksel değerleri tam sayı veya ikilik tabanda (binary) ifade edilebilir. İkilik tabanda ifade edildiğinde, en soldaki bite en önemli bit (MSB) ve en sağdaki bite de en önemsiz bit (LSB) denir.

Resim 2.4'de gri seviyeli bir fotoğrafta MSB ve LSB'ler vurgulanmıştır. En soldaki orijinal resmin piksel değerlerinin sadece MSB'leri tutularak diğer bitleri sıfırlandığında ortadaki, sadece LSB'leri tutularak diğer bitleri sıfırlandığında ise sağdaki görüntü elde edilmektedir [10].

Resim 2.4. MSB ve LSB'lerin belirgin görünümü [10]

Yük verisi ikilik taban yerine daha yüksek tabanlı bir sayıya dönüştürülerek de piksel değerlerine gömülebilir. Bu yöntem MBNS (Multiple Base Notational System) olarak adlandırılmaktadır [30].

PVD (Pixel Value Differentiation) tekniğinde taşıyıcı nesne bloklara bölünerek, blokların bağlantı noktalarındaki piksel fark değerleri değiştirilir [31].

Daha çok filigranlarda kullanılan QIM (Quantization Index Modulation) tekniğinde ise, yük bitleri nicel indeksleme yapılarak taşıyıcı resmin piksel değerlerine gömülür [32].

PBS (Prediction Based Steganography) tekniğinde, dosya sıkıştırma işlemlerindeki tahmin hatası değerleri (PEV - Prediction Error Value) kullanılır. Piksel değerleri yerine PEV'lerde değişim gerçekleştirilir [8].

Dönüşüm tabanlı (transform domain) teknikler

Ayrık Kosinüs Dönüşümü (DCT - Discrete Cosine Transform), görüntü işleme alanında sık kullanılan tekniklerden biridir. Ayrıca JPEG formatındaki görüntü dosyalarında da kayıplı sıkıştırma (lossy compression) tekniği olarak DCT kullanılmaktadır.

DCT tekniği ile elde edilen frekans sabitleri üzerinde çeşitli değişimler yapılarak veri gizleme işlemi gerçekleştirilebilir. Bu yöntemi kullanan bazı steganografi uygulamalarına örnek olarak JSteg, JPHide, YASS, Outguess, MB ve F5 verilebilir [1,4,8,33]. Westfeld tarafından 2001'de geliştirilen F5 algoritması, Bölüm 2.3.2'de ayrıntılı açıklanmıştır.

Ayrıca dönüşüm tabanlı teknikler arasında Ayrık Dalgacık Dönüşümü (DWT - Discrete Wavelet Transform) da sayılabilir. DWT'de resmin renk dalgacıkları 4 alt banda bölünür. En alt dalgacık bandı en önemli bilgileri içermektedir. JP2 isimli steganografi uygulaması bu tekniği kullanmaktadır.

Spektrum genişletme teknikleri

Bu teknikte taşıyıcı nesnenin gürültü (noise) özelliğinden faydalanılarak veri gizleme yapılır. Yüke spektrum genişletme (spread spectrum) uygulanarak taşıyıcı nesnenin gürültüsü içine gömülür. Sinyal gürültü oranının (SNR - Signal to Noise Ratio) düşük olduğu durumlarda fark edilemezlik artar. Spektrum genişletmeli resim steganografisi (SSIS - Spread-Spectrum Image Steganography) teknikleri sayesinde metin, görüntü veya başka herhangi bir sayısal sinyalin bir taşıyıcı resme gizlenmesi mümkündür [34].

Model tabanlı teknikler

Adaptif steganografi de denen bu tekniklerde, bağıl olasılık ve istatistiksel hesaplamalar kullanılır. Taşıyıcı nesnenin istatistiksel modeli çıkartılarak, fark edilemezliğin korunması şartıyla ne kadar veri gizlenebileceği, yani maksimum kapasite hesaplanır. Model tabanlı steganografi teknikleri oldukça genel kapsamlıdır ve hemen her türden taşıyıcı ortama uygulanabilirler [35].

2.2.4. Ağ (network) steganografisi

Veri gizleme üzerine yapılan çalışmalarda kullanılan yöntemlerden en yenisi ağ steganografisidir. Ağ steganografisinin temeli, OSI referans modelindeki protokollerin amaca uygun istismar edilmesi (exploitation) esasına dayanır. Bu tür steganografi uygulamaları, bir veya birden fazla protokolü aynı anda kullanır ya da protokoller arasındaki ilişkilerden faydalanır.

TCP/IP protokolünde bulunan bazı zafiyetler kullanılarak, iyi huylu görünen paketler ile gizli veri iletişimi gerçekleştirilebileceği ortaya konmuştur [36]. Bu gizli iletişim, TCP'nin başlangıçtaki el sıkışma (handshake) sekansında gerçekleştirilmektedir.

İnternet Protokolü üzerinden Ses (VoIP – Voice over Internet Protocol) uygulamaları da ağ steganografisinde kullanılabilmektedir. VoIP'in altyapısını oluşturan RTP ve UDP protokollerinde iletilen veri paketleri, mesajın gizlenebilmesine olanaklı yapıdadır [6].

Bahsedilen protokollerde kullanılmayan veya rezerve edilmiş paket bölümlerine veri gizlenerek uygulanan steganografi tekniklerinin yanı sıra, belirli ortam veya hizmetleri hedef alan steganografi uygulamaları da geliştirilmiştir. Skype veya BitTorrent gibi P2P dosya paylaşım hizmetleri, Facebook ve benzeri sosyal medya siteleri, WLAN, LTE ve bulut (cloud) gibi kablosuz ağ ortamları üzerinde steganografik yöntemler ile gizli veri iletimi yapılabilir [4]. Günümüzde kullanılan SRAC (Short Range Agent Communication) cihazları tarafından gönderilen paket serilerine veri gizlenerek, bu paketlerin alıcı tarafında birleştirilmesi ile de steganografi gerçekleştirilebilir [6].

2.3. Resim Steganografisinde Yaygın Kullanılan Algoritmalar

İletişim teknolojilerindeki gelişmeler sayesinde kişiler arası dosya paylaşımı her geçen gün kolaylaşmakta ve yaygınlaşmaktadır. Resim dosyalarının, iletişim kanalının güvenliği gözetilmeksizin yoğun bir şekilde paylaşılması, bu ortamın steganografik amaçlı kullanımını uzun süredir popüler kılmaktadır. Sıradan resim dosyalarına veri gizlenerek istenen alıcılara dikkat çekmeyecek şekilde iletilmesini sağlayan pek çok algoritma ve uygulama geliştirilmiştir.

Dijital resim dosyalarına veri gizleme teknikleri söz konusu olduğunda en sık başvurulanlar uzamsal ve dönüşüm tabanlı tekniklerdir. Uzamsal tabanlı tekniklerde temel sayılan LSB algoritması, günümüzde amaca uygun şekilde modifiye edilerek halen kullanılabilmektedir. Dönüşüm tabanlı teknikler arasında ise ki-kare saldırısına karşı fark edilemezlik özelliğini koruyabilmesi sebebiyle F5 algoritması öne çıkmaktadır.

2.3.1. LSB algoritması

Chan ve Cheng; bir imaj dosyasındaki en önemsiz bitler değiştirildiğinde gözle görünür bir fark meydana gelmeyeceğini göstermişlerdir [37]. LSB değiştirme (substitution) olarak adlandırılan bu yöntem gri seviyeli (grayscale) resimlere uygulanabileceği gibi, piksellerinde renk değeri bulunan resimlere de uygulanabilmektedir.

Taşıyıcı olarak seçilen 8 bitlik gri seviyeli bir resim C ile temsil ediliyor olsun. Bu taşıyıcı resim X_c eninde ve Y_c boyunda ise X_c x Y_c adet piksele sahiptir. Her bir piksel c_{ij} şeklinde temsil edilirse taşıyıcı resim;

$$C = \{ c_{ij} \mid 0 \leq i < X_c, 0 \leq j < Y_c, c_{ij} \in \{0, 1, \dots, 255\}\} \quad (2.1)$$

şeklinde ifade edilebilir. Eşitlik 2.1'de 0 ile 255 arasındaki değerler, c_{ij} pikselinin ikilik tabanda 8 hane (bit) ile yazılabilen gri seviye değeridir ($2^8 = 256$).

Uzunluğu n bit olan gizli mesaj M ile temsil edildiğinde ise aşağıdaki gibi yazılabilir;

$$M = \{ m_i \mid 0 \leq i < n, m_i \in \{0, 1\}\} \quad (2.2)$$

Eşitlik 2.2'deki m_i, M'nin tek bitini ifade eder.

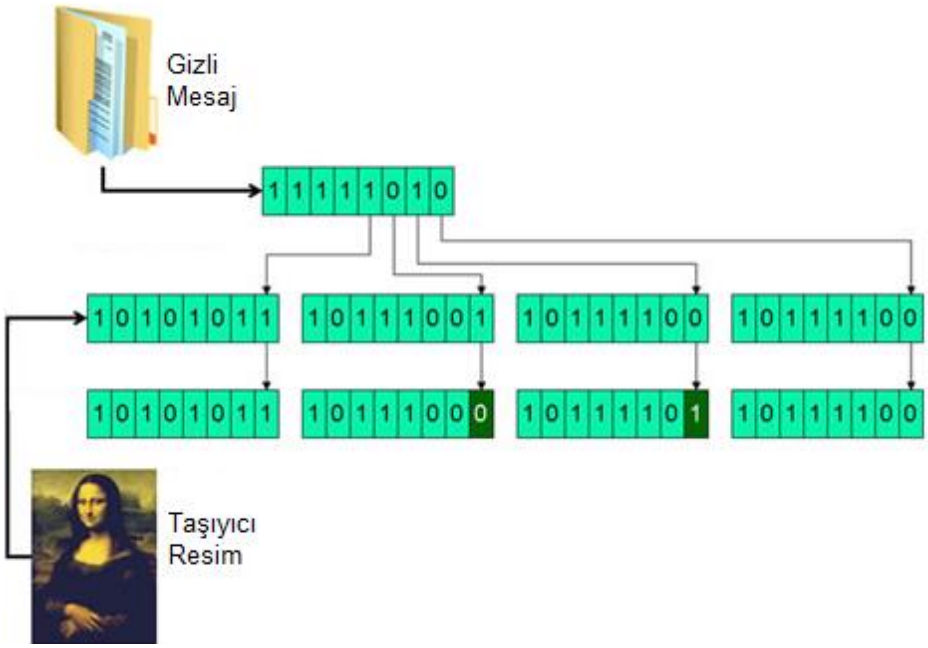
n < X_c x Y_c olmak üzere, tüm m_i'lerin taşıyıcı mesajdaki c_{ij} pikselinin son hanesi (LSB'si) ile değiştirilmesi sayesinde c'_{ij} stego-pikselleri, bunun sonucunda da Eşitlik 2.3'deki gibi ifade edilen C' stego-resmi elde edilir.

$$C' = \{ c'_{ij} \mid 0 \leq i < X_c, 0 \leq j < Y_c, c'_{ij} \in \{0, 1, \ldots, 255\}\} \quad (2.3)$$

Gizlenen mesajı tekrar elde etmek için veri ayıklama (extraction) işlemi uygulanırken ise, C' stego-resminde her pikselin son hanesi, mesaj uzunluğu sayısınca kaydedilir.

Gri seviyeli yerine renk değerlerine sahip piksel içeren resimler için de LSB algoritması aynı yöntemle işletilir. Bu durumda her bir pikselde üç farklı renk için üç grup 8 bitlik veri bulunması dikkate alınmalıdır.

LSB algoritması kullanılarak bir resme veri gizlenmesi işlemi Şekil 2.6'da şematize edilmiştir.



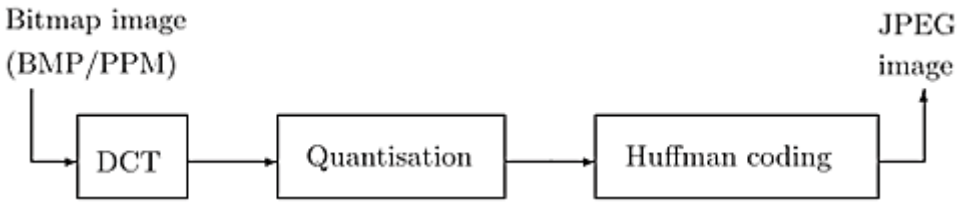
Şekil 2.6. LSB algoritması kullanılarak bir resme veri gizlenmesi

2.3.2. F5 algoritması

Andreas Westfeld tarafından 2001 yılında geliştirilen F5 algoritması, dönüşüm tabanlı steganografi teknikleri arasında halen en yaygın kullanıma sahiptir. Bu algoritma, JPEG kodlama sürecinin aşamaları arasında DCT katsayılarına müdahalede bulunarak veri gizlemeyi gerçekleştirir.

F5, yine bir dönüşüm tabanlı algoritma olan Jsteg üzerine iyileştirmeler getirmiştir [33]. En belirgin iyileştirmeler ise sıralamalı ayrım (permutative straddling) ve matris kodlama (matrix encoding) özellikleridir. Bu özellikler, Ron Crandall tarafından önerilen dağıtma fonksiyonları (spreading functions) ve matris kodlama tekniklerinden türetilmiştir[1].

JPEG görüntü kodlama tekniği, kayıplı sıkıştırma yapılmış resim dosyaları üretir. JPEG biçimindeki bir dosyada resim verileri, nicelendirilmiş (quantised) frekans katsayıları olarak tutulmaktadır. Bunu elde etmek için ilk önce sıkıştırılmamış görüntü (bit eşlem resmi) 8x8 piksellik bölümlere ayrılır. Daha sonra DCT uygulanarak piksellerin parlaklık (brightness) değerlerini içeren 8x8 matris, frekans katsayılarını içeren 8x8 matrise dönüştürülür. Bu aşamadaki frekans katsayıları reel sayılardır. Niceleme (quantisation) işlemi ile reel katsayılar, -2048 ile 2047 arasındaki en yakın tam sayı değerlerine yuvarlanır, dolayısıyla bu adımda kayıp oluşur. Tam sayı halindeki katsayı değerleri, Huffman kodlama yöntemi ile en az tekrar içerecek şekilde kodlanır ve JPEG resmi elde edilmiş olur. Şekil 2.7'de JPEG kodlamanın blok şeması verilmiştir [33].



Şekil 2.7. JPEG kodlamanın blok şeması [33]

[1] Crandall, R. (1998, Aralık). *Some Notes on Steganography.* Dresden Teknik Üniversitesi İşletim Sistemleri (TUD:OS - Technische Universität Dresden Operating Systems) Grubu'nun steganografi ile ilgili toplu posta listesine gönderilen çalışma, Dresden, Almanya.

Sıkıştırmadaki kaybın meydana geldiği niceleme adımı ile elde edilen katsayı değerlerinin (tam sayı) oluşma frekansları dağılımı incelendiğinde iki karakteristik özellik ortaya çıkmaktadır:

1. Katsayıların mutlak değerleri arttıkça, oluşma frekansları azalmaktadır. Yani, elde edilen katsayı değerleri arasında mutlak değeri büyük olanlar miktar olarak azdır. En yüksek oluşma frekansına sahip katsayı değeri ise 0'dır.

2. Mutlak değer artışına bağlı olarak oluşma frekanslarındaki azalma miktarı azalmaktadır. Yani, 0 ile 1 katsayılarının oluşma frekansları arasındaki fark, 1 ile 2 katsayılarının oluşma frekansları arasındaki farktan fazladır.

DCT tabanlı Jsteg algoritmasında veri gömme işlemi, elde edilen katsayı tam sayı değerlerinin ikilik tabana dönüştürülerek LSB'leri üzerine (matristeki sıra takip edilerek) mesaj bitlerinin yazılması ile gerçekleştirilir. Tüm mesaj bitleri katsayılara gömüldükten sonra Huffman kodlama ile devam edilerek JPEG stego-resmi elde edilir. Katsayılarda yapılan değişiklik sonucunda ise katsayıların oluşma frekansları dağılımı etkilenmekte ve bahsedilen iki karakteristik özellik bozulmaktadır. Westfeld, bu bozulmanın önüne geçerek karakteristik özelliklerin korunumunu sağlamak amacıyla sırasıyla F3, F4 ve F5 algoritmalarını geliştirmiştir [33].

F3 algoritmasının Jsteg'ten farkı, LSB'lerin üzerine yazmak yerine farklı bir işlem gerçekleştirmesidir. Matris sırasına göre, katsayıların ikilik taban değerlerindeki LSB'leri ile mesaj bitleri karşılaştırılarak, eşleşme durumunda herhangi bir işlem yapılmazken eşleşme olmadığı durumda ise katsayının mutlak değeri 1 azaltılır. Değeri 0 olan katsayılar üzerinde işlem yapılmaz. F3 sonucunda, 0'dan farklı tüm katsayıların ikilik taban değerlerinin LSB'leri gizli mesaj ile eşleşir hale gelir. Ancak, değeri 1 ve -1 olan katsayılar ile değeri 0 olan mesaj bitinin karşılaştırıldığı durumlarda, katsayıların yeni değerleri 0 yapılmaktadır. Bu da, frekans dağılımında büzüşme (shrinkage) adı verilen olumsuz duruma sebep olmaktadır.

Bu sorunu çözmek için geliştirilen F4 algoritması, katsayıların ikilik taban değerlerindeki LSB'leri olduğu gibi kullanmak yerine farklı bir yöntem izlemektedir. Tüm negatif katsayıların steganografik değerleri, tersi kabul edilir. Böylece negatif çift katsayıların

ikilik taban değerlerindeki LSB'leri 0 yerine 1, negatif tek katsayıların ikilik taban değerlerindeki LSB'leri 1 yerine 0, pozitif çift katsayıların ikilik taban değerlerindeki LSB'leri 0 ve pozitif tek katsayıların ikilik taban değerlerindeki LSB'leri de 1 olarak karşılaştırma işlemine alınır. Bu şekilde büzüşme problemi çözülmüş ve karakteristik özellikler korunmuş olur.

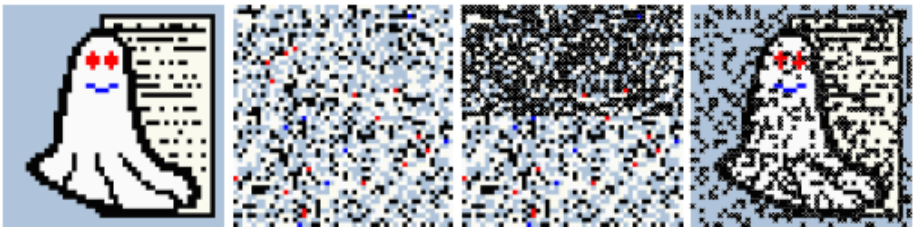
F5 algoritması ise, sıralamalı ayrım ve matris kodlama tekniklerini F4 üzerine uygulayarak daha iyi steganografik başarım elde edilmesini sağlamıştır.

Sıralamalı ayrım

Tipik bir uzamsal steganografi işlemi düşünüldüğünde, her mesaj bitinin taşıyıcıya gömülmesi adımında kullanılacak pikseller, taşıyıcının ilk pikselinden başlanarak sırayla seçilir. Bu gibi bir durumda eğer kapasitenin çoğu kullanılmamışsa, taşıyıcı resmin büyük bir bölümü değiştirilmeden kalacağı gibi, verinin gömüldüğü pikseller de resmin baş tarafında yığılmış halde bulunacaktır.

Olası herhangi bir steganaliz saldırısı ile daha iyi başa çıkabilmek için taşıyıcı resmin her bölümü mümkün olduğunca rastgele kullanılarak, veri gömme yoğunluğunun taşıyıcının her yerinde yakın değerde olması sağlanmalıdır. Dolayısıyla veri gömme adımlarında sıralı ilerlemek yerine, gömme işlemi yapılacak pikseller rastsal seçilmelidir.

Jsteg, F3 ve F4 algoritmalarında veri gömme işlemi her mesaj biti için, LSB'ler üzerine yazma veya sadece karşılaştırma fark etmeksizin, katsayıların matris sıralaması takip edilerek gerçekleştirilmektedir. F5 algoritması ise, girilen şifre sayesinde üretilen bir permütasyon işlemi kullanarak matristeki tüm katsayıların sırasını değiştirir. Yeni sıradaki katsayıların ikilik taban değerlerindeki LSB'leri mesaj bitleriyle karşılaştırılarak veri gömme işlemi tamamlandıktan sonra, permütasyon işlemi tersine doğru çalıştırılarak katsayıların matris sıralaması orijinal haline döndürülür. Böylece sıralamalı ayrım kullanılmış olup, katsayıların mutlak değerlerinde yapılan değişimlerin matrisin tümüne dağıtılması sağlanmaktadır. Resim 2.5'te sıralamalı ayrım sayesinde değişimlerin dağıtılması gösterilmiştir [33].



Resim 2.5. Sıralamalı ayrımın değişimleri dağıtması [33]

Matris kodlama

Bu teknik sayesinde, mesaj bitlerini taşıyıcıya gömme esnasında daha az sayıda değişiklik yapılabilmektedir. Tekdüze dağılıma sahip mesaj bitleri ile yine tekdüze dağılıma sahip taşıyıcı bitleri olduğu varsayılsın. Matris kodlama kullanılmayan standart bir uygulamada mesaj bitlerinin yarısı taşıyıcı bitlerinde değişiklik gerektirecek, diğer yarısı ise gerektirmeyecektir. Böylece her değişikliğe karşın 2 mesaj biti gömülmüş olacaktır. Matematiksel ifade ile gömme verimliliği (embedding efficiency) 2 bpc (bits per change) olacaktır. Çizelge 2.2'de verilen doğruluk tablosu ile de bu sonuç gösterilmiştir.

Çizelge 2.2. Tekdüze dağılımda 1 mesaj biti ile 1 taşıyıcı bitinin doğruluk tablosu

Mesaj biti m_1	Taşıyıcı biti c_1	Değişiklik
0	0	*Yok*
0	1	*Var*
1	0	*Var*
1	1	*Yok*

Toplam durum sayısı olan 4'te toplam sadece 2 adet değişiklik vardır. Dolayısıyla değişiklik olasılığı 1/2'dir. Gömülen mesaj biti sayısı 1 olduğuna göre, gömme verimliliği; 1 / (1/2) = 2 bpc'dir.

Matris kodlama kullanılmayan bir başka örnekte, mesaj biti ve taşıyıcı biti sayıları 2'ye çıkarılarak Çizelge 2.3'deki doğruluk tablosu elde edilmiştir. Bu örnekte, m_1'in c_1'e ve m_2'nin c_2'ye gömüleceği kabul edilerek, her ihtimal satırında kaç adet değişiklik yapılması gerektiği hesaplanmıştır. Taşıyıcı bitlerinden değişiklik yapılması gerekenler üstü çizili gösterilmiştir. Toplam durum sayısı olan 16'da toplam 16 adet değişiklik görülmektedir. Bir önceki örnekte kullanılan aynı hesaplama sonucunda, değişiklik olasılığı 16/16 ve

gömülen mesaj biti sayısı 2 olduğuna göre, gömme verimliliği yine; 2 / (16/16) = 2 bpc olarak bulunmaktadır. Mesaj biti sayısı artsa da, gömme verimliliği değişmemektedir.

Çizelge 2.3. Tekdüze dağılımda 2 mesaj biti ile 2 taşıyıcı bitinin doğruluk tablosu

Mesaj bitleri		Taşıyıcı bitleri		Değişiklik sayısı
m_1	m_2	c_1	c_2	
0	0	0	0	0
0	0	0	~~1~~	1
0	0	~~1~~	0	1
0	0	~~1~~	~~1~~	2
0	1	0	~~0~~	1
0	1	0	1	0
0	1	~~1~~	~~0~~	2
0	1	~~1~~	1	1
1	0	~~0~~	0	1
1	0	~~0~~	~~1~~	2
1	0	1	0	0
1	0	1	~~1~~	1
1	1	~~0~~	~~0~~	2
1	1	~~0~~	1	1
1	1	1	~~0~~	1
1	1	1	1	0
		Toplam=		16

Gömme verimliliğini arttırmak için matris kodlama tekniği kullanıldığında ise sonuç değişmektedir. İki mesaj biti (m_1 ve m_2) gömülmek istenen üç taşıyıcı biti (c_1, c_2 ve c_3) bulunduğu durumda, matris kodlama işlemi şu kurallara göre yapılır;

$m_1 = (c_1 \text{ XOR } c_3)$ VE $m_2 = (c_2 \text{ XOR } c_3)$ => değişiklik yapma
$m_1 \neq (c_1 \text{ XOR } c_3)$ VE $m_2 = (c_2 \text{ XOR } c_3)$ => c_1'i değiştir
$m_1 = (c_1 \text{ XOR } c_3)$ VE $m_2 \neq (c_2 \text{ XOR } c_3)$ => c_2'yi değiştir
$m_1 \neq (c_1 \text{ XOR } c_3)$ VE $m_2 \neq (c_2 \text{ XOR } c_3)$ => c_3'ü değiştir

Matris kodlama kullanılan durumun doğruluk tablosu Çizelge 2.4'te gösterilmiştir. Taşıyıcı bitlerinden değişiklik yapılması gerekenler üstü çizili gösterilmiş ve her ihtimal satırında kaç adet değişiklik yapılması gerektiği hesaplanmıştır. Toplam durum sayısı olan 32'de toplam 24 adet değişiklik görülmektedir. Önceki örneklerde kullanılan hesaplama sonucunda, değişiklik olasılığı 3/4 ve gömülen mesaj biti sayısı 2 olduğuna göre, gömme verimliliği; 2 / (3/4) = 2,67 bpc olarak bulunmaktadır.

Çizelge 2.4. Matris kodlama kullanıldığında doğruluk tablosu

Mesaj bitleri		Taşıyıcı bitleri			Değişiklik sayısı
m_1	m_2	c_1	c_2	c_3	
0	0	0	0	0	0
0	0	0	0	~~1~~	1
0	0	0	~~1~~	0	1
0	0	~~0~~	1	1	1
0	0	~~1~~	0	0	1
0	0	1	~~0~~	1	1
0	0	1	1	~~0~~	1
0	0	1	1	1	0
0	1	0	~~0~~	0	1
0	1	~~0~~	0	1	1
0	1	0	1	0	0
0	1	0	1	~~1~~	1
0	1	1	0	~~0~~	1
0	1	1	0	1	0
0	1	~~1~~	1	0	1
0	1	1	~~1~~	1	1
1	0	~~0~~	0	0	1
1	0	0	~~0~~	1	1
1	0	0	1	~~0~~	1
1	0	0	1	1	0
1	0	1	0	0	0
1	0	~~1~~	0	1	1
1	0	1	~~1~~	0	1
1	0	~~1~~	1	1	1
1	1	0	0	~~0~~	1
1	1	0	0	1	0
1	1	~~0~~	1	0	1
1	1	0	~~1~~	1	1
1	1	1	~~0~~	0	1
1	1	~~1~~	0	1	1
1	1	1	1	0	0
1	1	1	1	~~1~~	1
				Toplam=	24

F5 algoritmasında sıralamalı ayrım ve matris kodlama teknikleri uygulandıktan sonra, Huffman kodlama yapılarak JPEG stego-resmi oluşturulur. Elde edilen stego-resmin DCT katsayılarındaki değişim, sıralamalı ayrım kullanıldığı için dağıtılmıştır ve matris kodlama kullanıldığı için az sayıdadır.

3. STEGANALİZ

Mesaj iletimi gerçekleştirilebilecek herhangi bir ortamda gizlenmiş bir verinin var olup olmadığının ortaya çıkarılması amacıyla yapılan işlemlere steganaliz denir. Steganografi alanında "saldırı" olarak adlandırılan bu işlemler, kullanılan iletim ortamında gizli veriyi ekleme amacıyla yapılmış değişiklikleri belirlemeyi amaçlar [10].

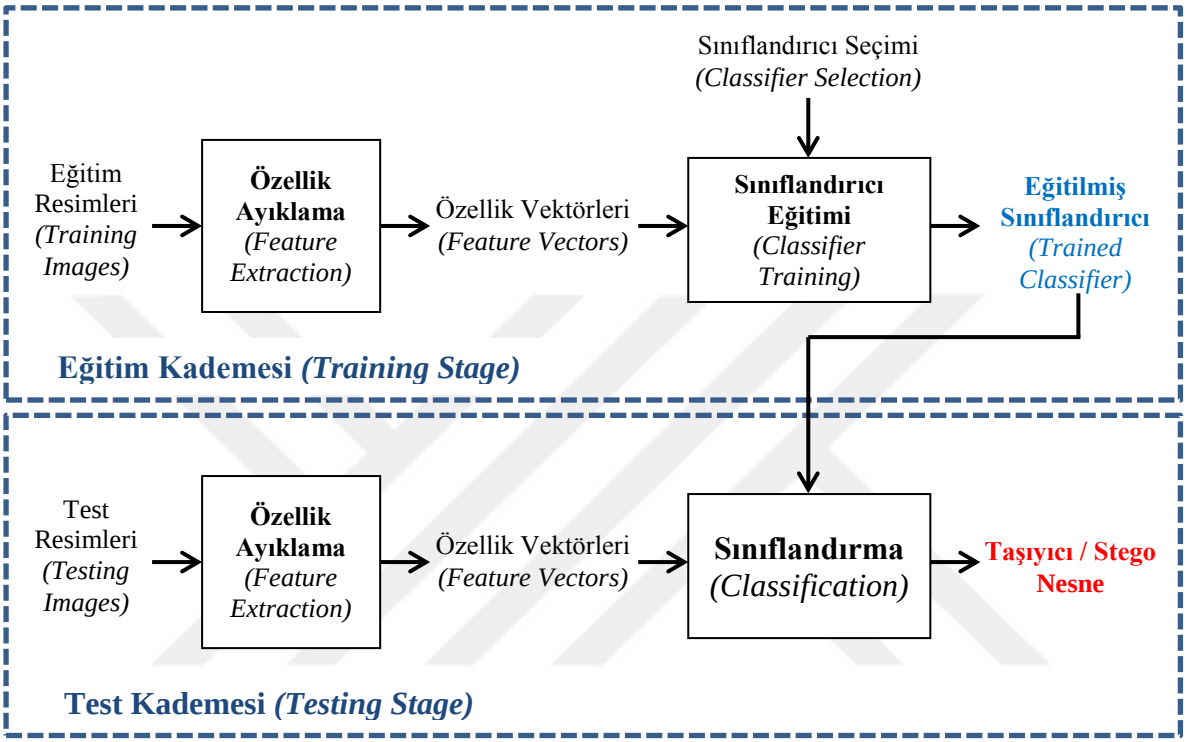
Bir kriptanaliz saldırısında gizli verinin varlığı bilinmekte ancak oluşturulmuş şifreleme yöntemi aşılmaya çalışılmaktayken, steganaliz saldırısında ise bir mesaja veri gizlenip gizlenmediği belirlenmeye çalışılır [9]. Gizlenmiş verinin varlığı belirlendiğinde, yani steganaliz başarılı olduğunda, gizli veriyi elde etme veya bozma/yok etme seçeneklerine sahip olunur. Bu durumda veriyi elde etme amacı taşınırsa işleme aktif steganaliz, alıcıya ulaşmaması için bozma veya yok etme seçeneği kullanılırsa pasif steganaliz denir [1].

Steganografik bir sistemde yük eğer şifrelenmiş ise, yükün fark edilmesi (detection) ve ayıklanması (extraction) için iki farklı işlem aşaması gerekir. Böyle bir durumda sistemin fark edilemezlik katmanını aşmak için steganaliz, güvenlik katmanını aşmak için ise kriptanaliz teknikleri kullanılır.

Steganaliz teknikleri çoğunlukla halihazırda bilinen steganografi uygulamalarına karşı özel olarak tasarlanır. Bu yönüyle steganaliz, tıpkı virüs tarama yazılımlarında olduğu gibi, sadece bilinen problemlere çözüm üretir [9]. Temel steganografik yöntemler sayıca çok olmamasına rağmen, uygulanış biçimindeki farklılaşma sonucunda birçok steganografi programı ortaya çıkmıştır. 2013 yılı itibariyle 200'ün üzerinde steganografi programı olduğu belgelenmiştir [6].

Steganalitik yaklaşımların temeli, aykırı durumların tespit edilmesine dayanır. Örneğin, benzer iki dosya arasındaki farkların, bir aykırılık tespitine imkan verecek şekilde ortaya çıkarılması işlemi bir steganaliz saldırısıdır. Steganalizi yapılmak istenen nesne ile mukayese edilecek benzer bir nesnenin mevcut olmadığı durumda, gerçekleştirilecek steganaliz işlemi *körleme saldırı* (blind attack) veya *evrensel steganaliz* olarak adlandırılır.

Tipik bir körleme saldırı yaklaşımında, eğitim ve test olmak üzere iki kademe içeren makine öğrenmesi tabanlı bir strateji kullanılır [8]. Eğitim kademesinin amacı, test kademesinde kullanılacak olan sınıflandırıcıyı elde etmektir. Şekil 3.1'de evrensel steganaliz işlem adımları gösterilmiştir.



Şekil 3.1. Evrensel steganaliz işlem adımları [8]

Westfeld ve Pfitzmann 2000 yılında yayınladıkları notlarda, çeşitli steganografik sistemlere görsel ve istatistiksel saldırılar gerçekleştirerek, bir resim verisindeki LSB'lerin tamamen önemsiz gürültü olarak kabul edilemeyeceğini ortaya koymuşlardır [38].

Fridrich ve Goljan, 2002'de yaptıkları çalışmada dijital resimler üzerinde görsel steganaliz, histogram analizi ve RS steganalizi gibi saldırılar gerçekleştirerek, hem LSB ekleme (addition) hem de LSB değiştirme (substitution) steganografi tekniklerinde fark edilemezliğin korunabilmesi için düşünüldüğünden daha az kapasite kullanımının gerektiğini göstermişlerdir [39].

Yine 2002 yılında Harmsen ve Pearlman tarafından yapılan çalışmada; LSB, DCT ve spektrum yayma yöntemleri analiz edilerek, gürültü katkılı veri gizleme modellemesi için bir çerçeve oluşturulmuştur [40].

2006'da Kharrazi, Sencar ve Memon, bilinen evrensel steganaliz yöntemlerini test etmek için JPEG dosyaları üzerinde uzamsal ve dönüşüm tabanlı steganografi teknikleri kullanmışlardır [41]. Çalışma sonucunda JPEG kalite faktörünün steganaliz performansını etkilediği ortaya konmuştur.

Westfeld, 2007'de yaptığı çalışmada steganaliz yöntemlerinin değerlendirme ölçütü olarak ROC eğrilerini kullanmıştır [42]. Bir ROC eğrisi, doğru pozitif oranlarının yanlış pozitif oranlarına göre değişimini gösterir.

Geetha, Sindhu ve Kamaraj tarafından 2009'da, steganografi uygulanan bir resim üzerinde bulunan içerikten bağımsız istatistiksel kanıtlar kullanılarak körleme saldırı yöntemi geliştirilmiştir [43].

Steganaliz yöntemleri geliştirilirken makine öğrenmesi tekniklerinden sıklıkla faydalanılmaktadır. 2010 yılında Liu, Sung, Qiao, Chen ve Ribeiro, JPEG resimlerinin steganalizi için destek vektör makinesi (SVM - Support Vector Machine) tekniklerinden faydalandıkları yeni bir yaklaşım geliştirmişlerdir [44]. Yine aynı yılda benzer bir çalışma gerçekleştiren Sabeti, Samavi, Mahdavi ve Shirani, yapay sinir ağları kullanarak piksel farklarına veri gömen steganografik sistemlerde steganaliz ile yük tahmini yapmışlardır [45]. 2013 yılında ise Li, Sencar ve Memon, geliştirdikleri karar ağacı tabanlı steganaliz yaklaşımı sayesinde, steganaliz tekniği seçimi sırasında tespit doğruluğundan ödün vermeden hesaplama maliyetini düşürmeyi başarmışlardır [46].

Genel olarak incelendiğinde temel steganaliz yaklaşımları; Görsel veya İşitsel Saldırılar, Yapısal Saldırılar ve İstatistiksel Saldırılar olmak üzere üç kategoride ele alınabilir [10].

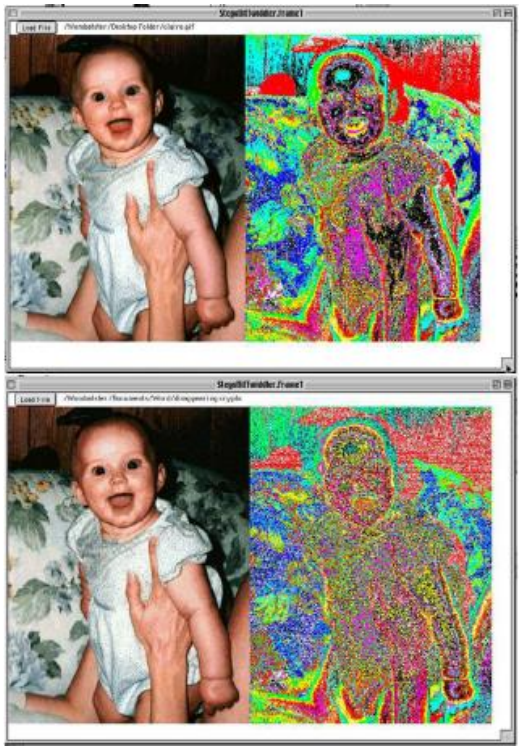
3.1. Görsel veya İşitsel Saldırılar

İnsanın görme veya işitme duyularını kullanarak bir mesajdaki anormalliği fark etmeyi amaçlayan steganaliz yaklaşımlarıdır. Bir resmin belirli bölümlerinin (örneğin yalnız LSB'lerinin) görünmesini sağlayacak araçlar kullanılıp gözle incelenerek bir anormallik olup olmadığı sonucuna varılabilir [10].

Görsel saldırı, bir görüntünün gömme katmanını (bir bit düzlemi gibi) tahmin etmek ve ardından bu katmandaki alışılmadık değişikliği aramak için katmanın görsel olarak incelenmesine dayanmaktadır. Bu yöntemler, stenografik gömme işleminin görüntüde neden olduğu küçük değişiklikleri ve/veya bozulmaları ortaya çıkarmayı amaçlar [47].

Orijinal resmin bilindiği durumlarda, şüphelenilen resimler ile orijinal taşıyıcı resimleri karşılaştırılarak analiz edebilir ve farklılıklar görünür hale getirilmeye çalışılabilir. Buna bilinen taşıyıcı saldırısı (known-cover attack) denir. Birçok görüntü karşılaştırıldığında, ortaya çıkan örüntüler bir steganografi aracının olası imzaları haline gelir. Bu imzalardan bazıları, gizli bilgilerin varlığını ve mesajları yerleştirmek için kullanılan araçları belirleyebilmektedir. Elde edilen bilgilerle, eğer karşılaştırma amaçlı orijinal taşıyıcı resimler mevcut değilse, bilinen imzalardan türetilmiş imzalar, bir gizli mesajın varlığını açığa çıkarmak ve mesaj gömmek için kullanılan aracı belirlemek için yeterli olabilmektedir [48].

Resim 3.1'in üst satırında taşıyıcı resim ve LSB'leri, alt satırında ise stego-resim ve LSB'leri görülmektedir. Taşıyıcı resim ve stego-resim arasında gözle görünür fark olmamasına rağmen LSB'ler incelendiğinde steganografi fark edilebilmektedir.



Resim 3.1. LSB'ler kullanılarak yapılan görsel steganaliz [10]

3.2. Yapısal Saldırılar

Gizli veri eklenmesi genellikle dosyanın yapısında fark edilebilir örüntüler oluşturacak şekilde değişime neden olur [10]. Steganografi için kullanılan bazı yazılımların yalnızca belli formatlarda veya boyutlarda dosya kullanımını desteklemesi, yapısal steganalizde fark edilemezliği azaltan bir etken olacaktır.

Yapısal steganaliz işleminde dosyanın içerik ve özellikleri dikkate alınmaktadır. Bu özelliklerden bazıları aşağıdaki gibidir:

- Dosya boyutundaki değişim
- Tarih/zaman bilgisinin kontrolü
- İçerik güncellenmesi
- Checksum bilgisi

Şüphelenilen bir dosyaya yapısal saldırı gerçekleştirmek için, dosya baytlarının değerlerini görüntülemeye yarayan metin düzenleyicisi (text editor) araçlarından faydalanılabilir. Resim 3.2'de bir dosyaya ait başlık bilgisinin kullanılmayan kısmına gizlenmiş olan veri ve bu verinin bir metin düzenleyicisi yazılımı aracılığıyla görüntülenmesi görülmektedir.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000F960	8D	6E	05	78	09	22	F2	B9	57	4B	A5	E6	75	83	5A	D8
0000F970	B7	DA	35	17	E4	75	6F	59	35	FC	69	DD	FD	A4	E9	30
0000F980	83	7F	8E	29	34	C4	1A	00	B5	93	41	79	B4	71	CD	C4
0000F990	4C	B7	5A	B7	1F	85	8F	01	4E	94	8D	9F	55	C8	18	C0
0000F9A0	10	59	30	F8	14	AC	97	EF	F1	FF	3A	B5	CA	82	3B	AF
0000F9B0	E7	FF	0F	50	4B	01	02	1E	03	14	00	02	00	08	00	F4
0000F9C0	02	9F	41	D4	50	D9	9C	E3	2D	00	00	99	2E	00	00	08
0000F9D0	00	18	00	00	00	00	00	00	00	00	00	A4	81	00	00	00
0000F9E0	00	20	20	20	20	20	20	20	20	55	54	05	00	03	8B	4B
0000F9F0	E1	50	75	78	0B	00	01	04	E8	03	00	00	04	E8	03	00
0000FA00	00	50	4B	05	06	00	00	00	00	01	00	01	00	4E	00	00
0000FA10	00	25	2E	00	00	00	00									

Resim 3.2. Bir dosyanın başlık kısmına gizlenen verinin görüntülenmesi

Görsel veya işitsel saldırılarda olduğu gibi, eğer orijinal dosya ulaşılabilir ise şüphelenilen dosya ile kolayca karşılaştırılabilmektedir. Gizli dosyaların görülmesini veya belirlenmesini sağlayan birçok araç mevcuttur. Windows Notepad gibi bir metin düzenleyicisi veya WnBrowse gibi bir hex editör aracı ile tutarsızlıklar belirlenebilir. Bir dosyanın içinde birden fazla dosya olup olmadığı, dosya imzalarındaki tutarsızlıktan

faydalanılarak steganaliz programları ile belirlenebilmektedir. Bazı örnek steganaliz programları ve özellikleri aşağıda verilmiştir.

WinHex

- ASCII - Hex dönüşümleri yapabilir.
- Dosyaların ASCII olarak birbirleri ile karşılaştırılmasını, farklı baytların bulunmasını ve raporlanmasını sağlar.
- Dosya işaretleyicisi yeteneklerine sahiptir.

StegSpy

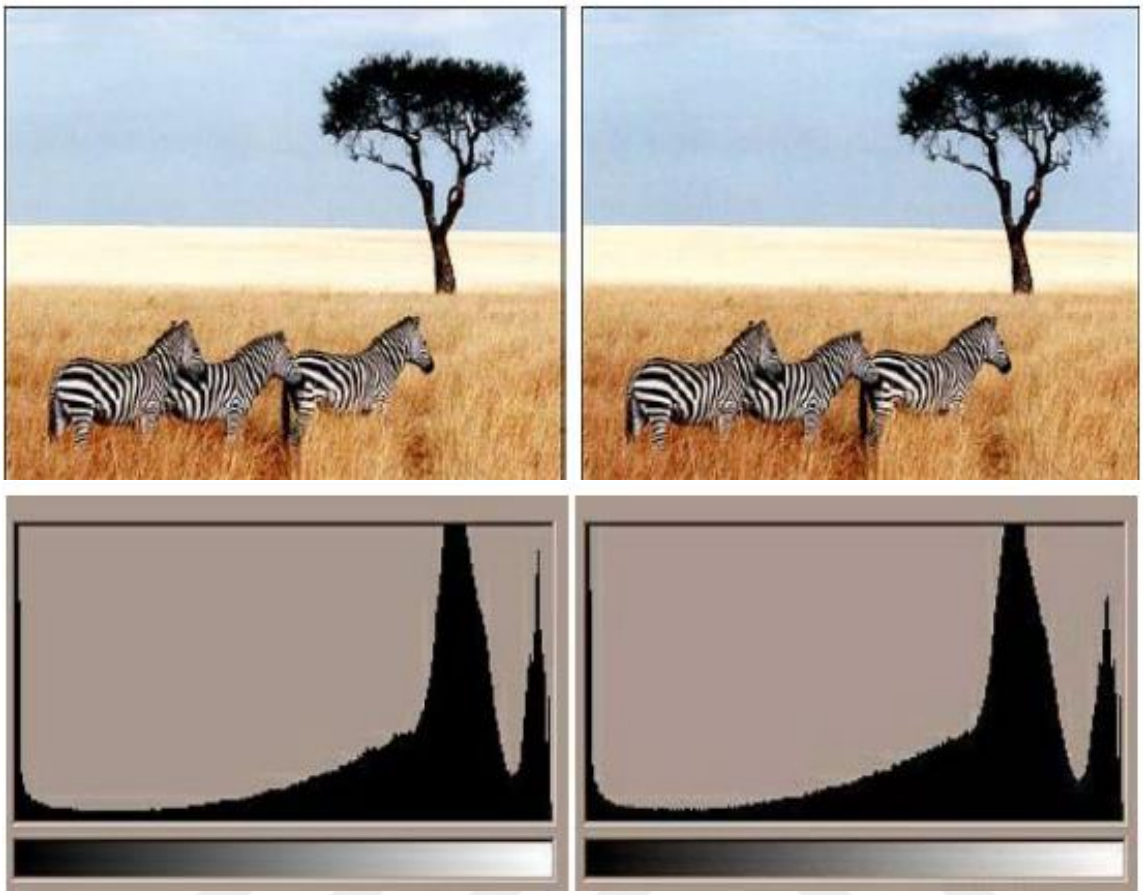
- Bilinen 13 farklı steganografi programına ait steganografik imzaları tanıyabilir.
- Mesajın saklı olduğu konumu tespit edebilir.
- Dosya ve dizinlerde tarama yapabilir.

3.3. İstatistiksel Saldırılar

Bir stego-nesnenin istatistikleri, veri gizleme nedeniyle değişime uğramıştır. Bir mesajdaki bitlerin veya bir resimdeki piksellerin istatistiksel profilinin incelenmesi sonucunda orijinal dosyaya müdahale olup olmadığı, dolayısıyla gizlenmiş veri bulunma ihtimali anlaşılabilir. İstatistiksel steganaliz, gömülü bilgileri tespit etmek için bir resmin temel istatistiklerini analiz eder [38, 49].

Çoğu steganografik sistem, bilinen parmak izlerini bir biçimde stego-nesnede bırakabilmektedir. İnsanlar tarafından algılanamasalar dahi, bu tür anormallikler kapsamlı bir istatistiksel saldırı sonucunda açığa çıkarılabilmektedir [50].

İstatistiksel steganaliz yaklaşımında histogram gibi birinci dereceden istatistiksel özellikler incelenebileceği gibi ikinci dereceden özelliklerde herhangi bir sapma olup olmadığı da incelenebilir [1]. Resim 3.3'de, bir taşıyıcı resme veri gizlenmeden önce ve veri gizlendikten sonra histogramda meydana gelen değişim görülmektedir.



Resim 3.3. Veri gizlemenin histograma etkisi [11]

LSB steganografisinin özellikleri göz önüne alındığında, istatistiksel analiz aracı olarak farklı görüntü histogramları seçilebilmektedir. I resminin (i, j) konumundaki yoğunluğunun I(i, j) olduğu kabul edilsin. Farklı bir D resmi; D(i, j) = I(i + 1, j) - I(i, J) olarak alınsın. Bu durumda genel olarak, farklı resmin genelleştirilmiş bir Gauss dağılımını izlediği kabul edilir [51]. Buna ilişkin olasılık yoğunluk fonksiyonu şu şekilde yazılabilir:

$$p_{v,\beta}(x) = \frac{v}{2\beta\Gamma(1/v)} \exp\left\{-\left(\frac{|x|}{\beta}\right)^{v}\right\} \tag{3.1}$$

Eşitlik 3.1'de *v*; şekil faktörü olarak adlandırılmaktadır. Hem Gauss dağılımları hem de Laplace dağılımları sırasıyla şekil faktörleri 2 ve 1 olan genelleştirilmiş Gauss dağılımının özel bir örneği olarak belirtilir [52].

3.3.1. Ki-kare testi

Gözlenen değerlerin, teorik (beklenen) değerlere uygun olup olmadığını belirlemeye yarayan istatistiksel yönteme ki-kare testi denir [53]. Ki-kare testi, bir veri setinin düzgün dağılm gösterip göstermediğini aşağıdaki eşitlik ile ortaya çıkarır:

$$\chi^2 = \sum \frac{(o_i - e_i)^2}{e_i} \tag{3.2}$$

Eşitlik 3.2'de o_i gözlenen sıklık değerini, e_i ise beklenen değeri temsil eder. İşlem sonucu elde edilen χ^2 (ki kare) değeri ne kadar yüksek ise gözlenen değer olması gerekene o kadar yakın demektir.

Ki-kare testi dijital dosyaların steganalizi için kullanılmak istendiğinde, Eşitlik 3.2'deki sıklık değişkenleri, dosyanın bit yapısındaki 1'lerin veya 0'ların görülme sıklığını (frekansını) ifade edecek şekilde düzenlenir.

Veri gizlenmemiş bir resimde LSB'lerin frekansları birbirlerinden çok farklı değerlerdedir. LSB'leri değiştiren bir steganografi işlemi gerçekleştirildiğinde ise, ardışık bit konumlarında değer çiftleri (pairs of values) oluşmaya başlar. Bu durumda beklenen frekans değeri (e_i), her gözlenen bit çiftinin frekans değeri için bu çiftlerin frekans değerlerinin ortalamasına eşit olacaktır [38].

8 bitlik bir resim dosyasının her biriminde (piksel veya renk) bulunan 256 farklı değer olduğu için, hesabın serbestlik derecesi (degree of freedom) 255 alınır (k – 1). Böylece bulunan ki-kare değeri ile gama fonksiyonu kullanılarak Eşitlik 3.3'de görüldüğü gibi ki-kare dağılımlı olasılık yoğunluk fonksiyonunun p değeri elde edilir:

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma\left(\frac{k-1}{2}\right)} \int_0^{\chi^2_{k-1}} e^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} dx \tag{3.3}$$

4. K5 STEGANOGRAFİ UYGULAMASI

Resim steganografisi çalışmalarında dönüşüm tabanlı yöntemlere doğru bir eğilim söz konusu olsa da, uzamsal tabanlı çalışmalara halen yenileri eklenmektedir. Bunun başlıca sebepleri, uzamsal tabanlı yöntemlerde uygulanabilirliğin dönüşüm tabanlı yöntemlere kıyasla daha az karmaşıklık gerektirmesi ve farklı yaklaşımların bir arada kullanımının mümkün olmasıdır.

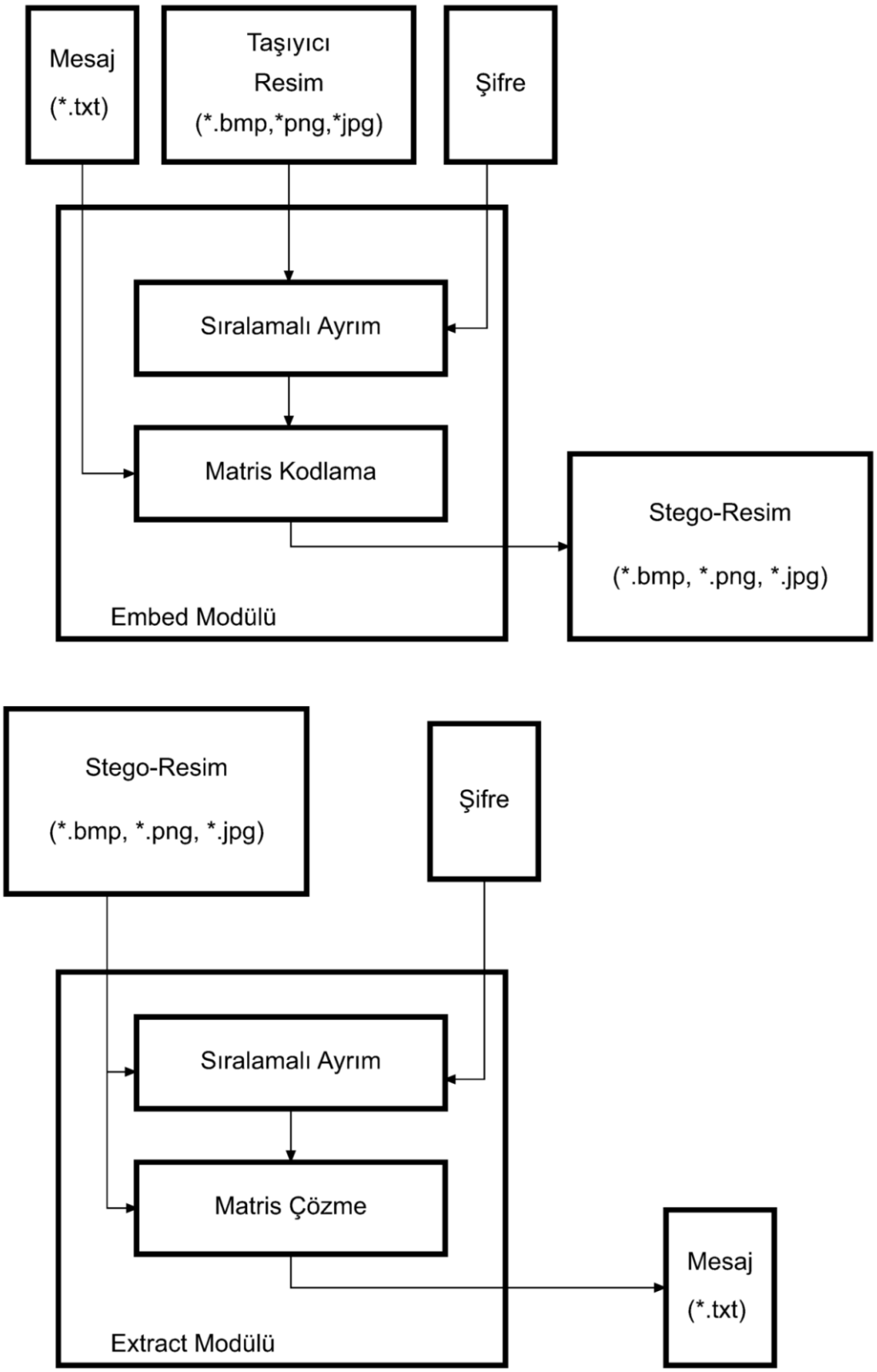
Önerilen K5 Steganografi uygulamasının model aldığı F5 algoritması da Jsteg gibi dönüşüm tabanlı çalışmaktadır ve JPEG resim kodlama aşamaları sırasında veri gizleme işlemini gerçekleştirir. Dolayısıyla F5 algoritması sonucunda yalnızca JPEG biçiminde stego-resim elde edilebilmektedir. Bu biçim kısıtından kurtulabilmek için, steganografik işlemin dönüşüm tabanı yerine uzamsal tabanda gerçekleştirilmesi sağlanabilir.

K5 Steganografi uygulaması, sıralamalı ayrım ve matris kodlama işlemlerini içermesi itibariyle F5 ile kavramsal benzerlik göstermektedir. Bununla birlikte, veri gizleme için uzamsal tabanlı steganografi yöntemleri kullanılması sebebiyle uygulamada F5'ten tamamen farklılaşmıştır. Dönüşüm tabanlı yerine uzamsal tabanlı çalışması sayesinde K5 Steganografi uygulaması, çok biçimli resim desteği sağlamaktadır. K5 Steganografi uygulamasının blok şeması Şekil 4.1'de gösterilmiştir.

4.1. Yöntem ve Kullanılan Araçlar

K5 Steganografi uygulamasını geliştirmek için Microsoft .NET Framework (sürüm 4.6) platformunda Microsoft Visual Studio 2012 (sürüm 11.0.61219, güncelleme 5) programı altında Visual C# 2012 programlama dili kullanılmıştır (kısaca C# olarak anılacaktır).

Beşinci bölümde gerçekleştirilen steganaliz saldırıları için ihtiyaç duyulan istatistiksel hesaplamaları yapmak üzere C# programlama diline ALGLIB kütüphanesinin 3.10 sürümü eklenmiştir [54]. Ayrıca yine aynı bölümde karşılaştırma amacıyla elde edilmesi gereken stego-resimler için VSL Tool yazılımının LSB Encoding ve F5 Encoding modülleri kullanılmıştır [55].



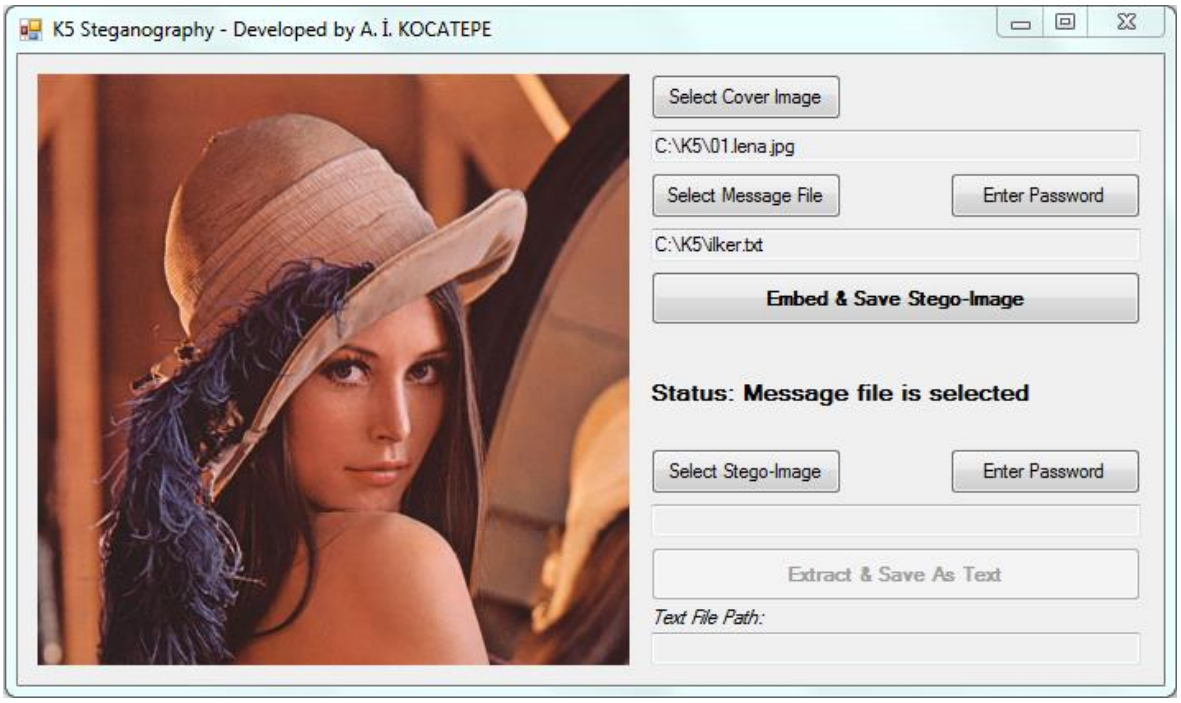
Şekil 4.1. K5 Steganografi uygulamasının blok şeması

4.2. K5 Uygulamasının Girdileri ve Çıktıları

Geliştirilen K5 Steganografi uygulamasının giriş arayüz ekran görüntüsü Resim 4.1'de görülmektedir. Uygulama, Veri Gömme (Embedding) ve Veri Ayıklama (Extraction) olmak üzere iki modül halinde çalışmaktadır. Ayrıca ana ekranda her iki modülü ayıran kısımda, uygulamanın anlık durumunu metinle bildiren bir etiket (label) bulunmaktadır. Uygulama ilk kez çalıştırıldığında bu etikette "Status: Ready" metni görünür.

Veri gömme modülündeki algoritmalar için gerekli olan girdiler; taşıyıcı resim, gizlenecek mesaj ve şifre metnidir. Bu aşama sonucunda uygulamanın ürettiği çıktı ise stego-resimdir. İlgili modülde kullanılan metotların C# kaynak kodları Ek-1'de verilmiştir.

Veri ayıklama modülünde ise stego-resim ve şifre metni olmak üzere iki girdi bulunmaktadır. Bu modüldeki şifre metni, veri gömme modülündeki şifre metninden farklı bir nesnede tutulmaktadır. Ancak ileride de anlatılacağı gibi, ayıklama şifresi ile gömme şifresinin farklı değerlerde olması sonucunda gizlenen mesaj elde edilemeyecektir. Veri ayıklama modülünün çıktısı ise mesaj metin dosyası olmak üzere bir tanedir. Bu modülde kullanılan metotların C# kaynak kodları Ek-2'de verilmiştir.

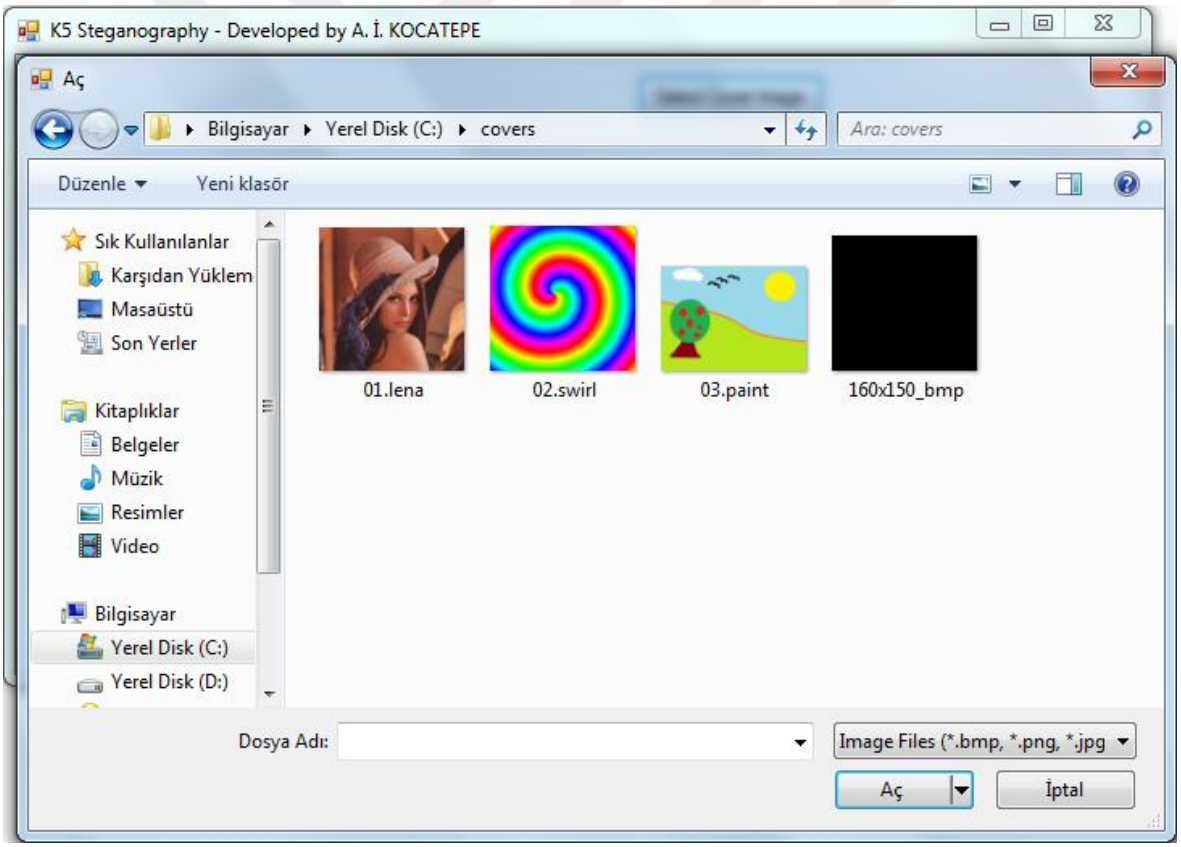


Resim 4.1. K5 Steganografi uygulamasının giriş arayüz ekranı

4.2.1. Taşıyıcı resim

"Select Cover Image" butonu tıklandığında, kullanıcıya bir dosya açma ekranı (OpenFileDialog) gösterilir. Bu ekran kullanılarak seçilebilecek dosya türleri "Image Files" ve uzantıları *.bmp, *.png ve *.jpg olarak filtrelenmiştir. Taşıyıcı resim seçimi için dosya açma ekranı Resim 4.2'de görülmektedir.

Filtrelemeye uygun bir dosya seçilerek "Aç" (veya "Open") tuşuna basıldığında, "Select Cover Image" butonunun altındaki metin kutusunda dosya yolu görülecektir. Ayrıca ana ekranın sol yarısında bulunan resim kutusu da seçilen resmin sığdırılmış (stretched) halini gösterecektir.



Resim 4.2. Taşıyıcı resim seçimi için dosya açma ekranı

Eğer taşıyıcı resim başarılı bir şekilde seçilmişse, uygulama tarafından "Embed & Save Stego-Image" butonunun aktifleştirilmesini sağlayan bileşenlerden biri olan *coverEtkin* mantıksal değişkeni, "true" değerini alacaktır. Etiket metninde ise "Status: Cover Image Selected" durumu yazacaktır.

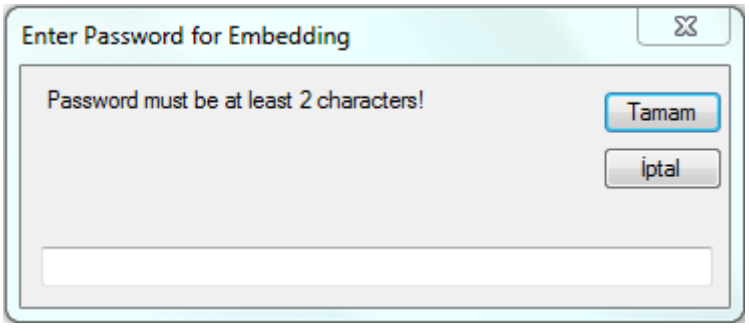
4.2.2. Gizlenecek mesaj metni

"Select Message File" butonu tıklandığında, kullanıcıya bir dosya açma ekranı (OpenFileDialog) gösterilir. Bu ekran kullanılarak seçilebilecek dosya türü "Text Files" ve uzantısı *.txt olarak filtrelenmiştir. Filtrelemeye uygun bir dosya seçilerek "Aç" (veya "Open") tuşuna basıldığında, "Select Message File" butonunun altındaki metin kutusunda dosya yolu görülecektir.

Eğer mesaj metnini içeren dosya başarılı bir şekilde seçilmişse, dosya içerisinde yazılı metnin tamamı "ReadAllText" komutu ile okunarak, public string değişken olan *gizlimesaj*'a aktarılır. Ayrıca uygulama tarafından "Embed & Save Stego-Image" butonunun aktifleştirilmesini sağlayan bileşenlerden biri olan *mesajEtkin* mantıksal değişkeni, "true" değerini alacaktır. Etiket metni ise "Status: Message File Selected" durumunu gösterecektir.

4.2.3. Şifreler

"Enter Password" butonlarından herhangi biri tıklandığında, kullanıcıya bir metin giriş kutusu (InputBox) gösterilir. Bu komutun C# dilinde kullanılabilmesi için uygulama projesine Microsoft Visual Basic kütüphanesi (Microsoft.VisualBasic.dll) eklenmiştir. Resim 4.3'te şifre giriş ekranı görülmektedir.



Resim 4.3. Şifre giriş ekranı

Kullanıcı tarafından girilen şifre metni uzunluğu iki karakterden az ise, kullanıcının şifreyi yeniden girmesi için metin giriş kutusu tekrar gösterilir. Metin giriş kutusu vasıtasıyla kullanıcıdan alınan şifre, veri gömme işlemlerinde kullanılmak üzere *pswEmbed* isimli

public string değişkene aktarılır. Ayrıca uygulama tarafından "Embed & Save Stego-Image" butonunun aktifleştirilmesini sağlayan bileşenlerden biri olan *pswEmbedEtkin* mantıksal değişkeni, "true" değerini alacaktır.

Eğer girilen şifre veri gömme modülüne aitse etikette "Status: Embed Password OK" durumu, veri ayıklama modülüne ait olduğunda ise "Status: Extract Password OK" durumu gösterilecektir. Veri ayıklama modülü için girilen şifre *pswExtract* isimli public string değişkende saklanır. Bu durumda uygulama tarafından "Extract & Save As Text" butonunun aktifleştirilmesini sağlayan bileşenlerden biri olan *pswExtractEtkin* mantıksal değişkeni, "true" değerini alacaktır.

4.2.4. Stego-resim

Veri gömme modülünün çıktısı olan stego-resim, başarılı olan veri gömme işlemi sonrasında bir dosya kaydetme ekranı (SaveFileDialog) gösterilerek kullanıcıya kaydettirilir. Dosya kaydetme ekranında doğru işlemler uygulanarak stego-resim kaydedildiğinde veri gömme modülü sonlanmış olur ve etikette "Status: Embedding completed… Ready" metni görünür.

Veri ayıklama modülünün girdisi olan stego-resim ise, tıpkı diğer modüldeki taşıyıcı resim seçiminde olduğu gibi bir dosya açma ekranı sayesinde kullanıcıya seçtirilir. Bu dosya açma ekranında da dosya türleri "Image Files" ve uzantıları *.bmp, *.png ve *.jpg olarak filtrelenmiştir. Filtrelemeye uygun bir dosya seçilerek "Aç" (veya "Open") tuşuna basıldığında, "Select Stego-Image" butonunun altındaki metin kutusunda dosya yolu görülecektir. Ayrıca ana ekranın sol yarısında bulunan resim kutusu da seçilen resmin sığdırılmış (stretched) halini gösterecektir.

Eğer veri ayıklama modülünün çalışması sırasında stego-resim başarılı bir şekilde seçilmişse, uygulama tarafından "Extract & Save As Text" butonunun aktifleştirilmesini sağlayan bileşenlerden biri olan *stegoEtkin* mantıksal değişkeni, "true" değerini alacaktır. Etiket metninde ise "Status: Stego-Image Selected" durumu yazacaktır.

4.2.5. Ayıklanan mesaj metni

Veri ayıklama modülünün her iki girdisi de (şifre ve stego-resim) kullanıcı tarafından girildiğinde, “Extract & Save As Text” butonu aktif hale gelecektir. Bu aşamada gerçekleştirilecek olan ve Bölüm 4.4’te anlatılan işlemler sonucunda, veri ayıklama modülünün çıktısı olarak mesaj metni elde edilecek ve bir dosya kaydetme ekranı sayesinde kullanıcıya kaydettirilecektir. Bu adımdaki dosya kaydetme ekranı, dosya türlerini “Text Files” ve uzantıları *.txt olarak filtreler.

4.3. Veri Gömme (Embedding) Modülü

Veri gömme modülünün başlatılabilmesi için “Embed & Save Stego-Image” butonunun aktif olması gerekmektedir. Aktifleştirme işleminde, girdilerin tamamının etkin olup olmadığına bakılır. Her girdi metodu içerisinde tanımlı “Etkin” değişkenlerinin tamamı true değerinde ise, “Embed & Save Stego-Image” butonu aktif hale gelir.

Veri gömme metoduna, taşıyıcı resimden yeni bir resim türetilerek başlanır. Daha sonra mesaj metninden bitlerin elde edilmesi, sıralamalı ayrım ile *pswEmbed* kullanılarak veri gömülecek piksellerin yerlerinin değiştirilmesi, matris kodlama kullanılarak renk değerlerinin değiştirilmesi ve üretilen stego-resmin kaydedilmesi adımları uygulanır. Veri gömme modülüne ait C# kaynak kodları Ek-1’de verilmiştir.

4.3.1. Mesaj metninden bitlerin elde edilmesi

Bu adımda ilk önce, mesaj metninin kaydedildiği *gizlimesaj* isimli string değişkenin uzunluğu ölçülerek, bu uzunluk değeri sayısınca tekrar edecek döngü başlatılır. Döngünün her adımında bir mesaj harfi okunarak char türündeki değişkene kaydedilir. Burada kaydedilen değer, okunan harfin ASCII karşılığıdır.

Sonrasında, char değişkeninin ikilik tabanda gösterimi gerçekleştirilir ve *hrfbts* isimli string değişkene aktarılır. Böylece mesaj metnindeki her harften, 8 bitlik binary veri elde edilmiş olur. Dolayısıyla her bir harfi saklamak için taşıyıcı resimde 8 bitlik bölgeye ihtiyaç duyulacaktır. Bu bölge ise, matris kodlama da dikkate alındığında, taşıyıcı resmin 4 piksellerindeki R, G, B renk değerlerinden elde edilir.

Piksel renk değerlerinin her biri de 8 bitlik veriden oluşmasına rağmen yalnızca ilgili rengin LSB'si kullanılacaktır. Dört pikseldeki her üç renk değerinden ikisinin LSB'leri seçilerek toplam 8 bitlik harf verisini gömmeyi sağlayacak yer elde edilmiş olur.

4.3.2. Sıralamalı ayrım ile piksellerin seçimi

F5 algoritmasındaki katsayı değerlerinin sıralamasında yapılan değişikliğe benzer olarak, taşıyıcı resimdeki renk değerleri kullanılacak olan piksellerin seçimleri de orijinal piksel sırasını takip etmeyecek bir şekilde yapılır. Sıralamalı ayrım işlemi için şifre metni ile resmin en (*w*) ve boy (*h*) değerleri kullanılmaktadır.

Sıralamalı ayrım işleminde öncelikle şifre metninin karakterleri kullanılarak, *bits0* ve *bits1* adlarında iki string ifade elde edilir. 0 ve 1 karakterleri içeren bu binary stringler, resmin piksel haritasındaki sütun ve satırlara denk gelecek tamsayıları oluşturmak için, bir doğrusal geri-beslemeli kayan kaydedici (LFSR – Linear-Feedback Shift Register) benzeşiminde ilk değer (seed) olarak kullanılacaklardır.

LFSR metodu, sıfır olmayan ikilik tabandaki sayıların son iki hanesinin XOR sonuçlarını en sol haneye yazarak döngüsel bir şekilde ikilik tabanda sıra takip etmeyen sayılar üretir. LFSR ile üretilecek sütun sayılarının *w* değerini, satır sayılarının da *h / 4* değerini geçmemesi dikkate alınır. Boy değerinin dörde bölünmesinin sebebi, veri gömme modülünde her adımda gizlenecek olan 8 harf bitinin 4 piksel gerektirmesidir.

Sütun ve satırlar için üretilmiş sıra takip etmeyen tamsayılar, dizi değişkenler (*iarray* ve *jarray*) olarak sıralamalı ayrım metodunun çıktılarını oluştururlar. LFSR metodu ile sıfır değeri elde edilemediğinden, her iki dizinin de son elemanına "0" değeri atanır. Bu aşamadan sonra yapılacak işlemler, pikselleri orijinal sütun ve satır sayılarına göre sırayla seçmek yerine ilgili dizi değerlerinin belirttiği sütun ve satır sayılarına göre seçerek gerçekleştirilir.

4.3.3. Matris kodlama kullanılarak renk değerlerinin değiştirilmesi

Bu aşamaya gelindiğinde, 8 bitlik mesaj harf bitlerinin ikişerli grupları, ilgili pikselin R, G, B değerlerinin LSB'leri ile matris kodlama işlemine tabi tutulur. Aşağıdaki kurala göre renklerin LSB'lerinde değişiklik yapılarak veri gömülmüş olur.

h_1 = (R XOR B) VE h_2 = (G XOR B) => değişiklik yapma
$h_1 \neq$ (R XOR B) VE h_2 = (G XOR B) => R'yi değiştir
h_1 = (R XOR B) VE $h_2 \neq$ (G XOR B) => G'yi değiştir
$h_1 \neq$ (R XOR B) VE $h_2 \neq$ (G XOR B) => B'yi değiştir

Böylece her piksele iki harf biti gömülmüş olur. Dolayısıyla 8 bitlik bir harfi gizlemek için toplam dört piksel kullanılmaktadır. Sıralamalı ayrım, LFSR ve Matris kodlama metotlarının C# kaynak kodları Ek-2'de verilmiştir.

4.3.4. Stego-resmin kaydedilmesi

Piksellerinin renk değerlerinin LSB'lerine, mesaj metnindeki karakterlerin tamamı gizlenmiş olan taşıyıcı resim, kaydedilmeye hazır bir stego-resme dönüştürülmüş olur.

Stego-resmin kaydedilmesi, "Embed & Save Stego-Image" butonuna basıldığında başlayan veri gömme aşamasının son adımıdır. Bir dosya kayıt ekranı (SaveFileDialog) açılarak, kullanıcıya, stego-resmin kaydedileceği dizin seçtirilir. Stego-resme isim verilerek "Kaydet" (veya "Save") tuşuna basıldığında, uygulamanın veri gömme modülü tamamlanmış olur. Bu durumda etiket metni "Status: Embedding completed… Ready" olarak görünecektir.

4.4. Veri Ayıklama (Extraction) Modülü

Veri ayıklama modülünün başlatılabilmesi için "Extract & Save As Text" butonunun aktif olması gerekmektedir. Aktifleştirme işleminde, girdilerin tamamının etkin olup olmadığına bakılır. Her girdi metodu içerisinde tanımlı "Etkin" değişkenlerinin tamamı true değerinde ise, "Extract & Save As Text" butonu aktif hale gelir.

Veri ayıklama metoduna, seçilen stego-resimden yeni bir resim türetilerek başlanır. Daha sonra *pswExtract* ve sıralamalı ayrım kullanılarak stego-piksellerin bulunması, renk değerlerinden mesaj metni elde edilmesi ve mesaj metninin kaydedilmesi adımları uygulanır. Veri ayıklama modülüne ait C# kaynak kodları Ek-3'te verilmiştir.

4.4.1. Stego-resim seçimi

Stego-resim, bir dosya açma ekranı sayesinde kullanıcıya seçtirilir. Bu dosya açma ekranında da dosya türleri "Image Files" ve uzantıları *.bmp, *.png ve *.jpg olarak filtrelenmiştir. Filtrelemeye uygun bir dosya seçilerek "Aç" (veya "Open") tuşuna basıldığında, "Select Stego-Image" butonunun altındaki metin kutusunda dosya yolu görülecektir. Ayrıca ana ekranın sol yarısında bulunan resim kutusu da seçilen resmin sığdırılmış (stretched) halini gösterecektir.

Stego-resim başarılı bir şekilde seçilmişse, uygulama tarafından "Extract & Save As Text" butonunun aktifleştirilmesini sağlayan bileşenlerden biri olan *stegoEtkin* mantıksal değişkeni, "true" değerini alacaktır. Etiket metninde ise "Status: Stego-Image Selected" durumu yazacaktır.

4.4.2. Şifre ve sıralamalı ayrım kullanılarak stego-piksellerin bulunması

Veri ayıklama şifresi (*pswExtract*) kullanılarak, seçilmiş olan stego-resmin en ve boy değerleri de hesaba katılıp, veri gömme modülünde işlem gören sıralamalı ayrım (*permStrad*) metodu çalıştırılır.

Stego-resim ile taşıyıcı resmin en ve boy değerlerinin değişmediği kabul edilirse, *pswExtract* değişkeninin *pswEmbed* değişkenine eşit olması durumunda, veri gömme modülünde işlem gören piksellerin aynısı veri ayıklama modülünde de işlem görecektir. Bunun sebebi, LFSR metoduna girdi olarak verilecek değişkenlerin veri gömme modülündekiler ile aynı olmaları durumunda, aynı ilk değer (seed) sayesinde yine aynı tamsayı değerlerinin elde edilip sütun ve satır belirten dizi değişkenlerde saklanacak olmasıdır.

Bu adım sayesinde ilk önce mesaj uzunluğunun kaydedildiği piksellerin işlem görmesi sağlanarak, ayıklanacak gizli mesajın toplam bit uzunluğu hesaplanır.

4.4.3. Renk değerlerinden mesaj metni elde edilmesi

Sıralamalı ayrım sonucunda işlem görecek pikseller belirlenmiş olup, bu adımda ise bu piksellerin LSB'lerinden veri ayıklanması gerçekleştirilecektir. Veri gömme modülündeki matris kodlama işlemi sonucunda piksellerin R, G, B değerlerinin LSB'lerine yerleştirilen bitler, aşağıdaki kurala göre XOR işlemine tabi tutularak ayıklanır:

$m_1 = (R \text{ XOR } B)$ VE $m_2 = (G \text{ XOR } B)$

Bu şekilde her pikseldeki renk değerlerinin uygun XOR işlemleri sonucunda iki mesaj biti (m_1 ve m_2) elde edilmiş olur. Dolayısıyla her dört pikselden toplam 8 bitlik mesaj ayıklanır, bu da bir harf demektir.

Her harfin tüm bitleri (8 tanesi) tamamlandığında, ilgili harf *outtext* isimli string değişkenin sonuna eklenir (*outtext* başlangıçta boştur). Mesaj uzunluğu boyunca süren döngü tamamlandığında ise *outtext*'te gizli mesajın tamamı mevcut halde olacaktır.

4.4.4. Mesaj metninin kaydedilmesi

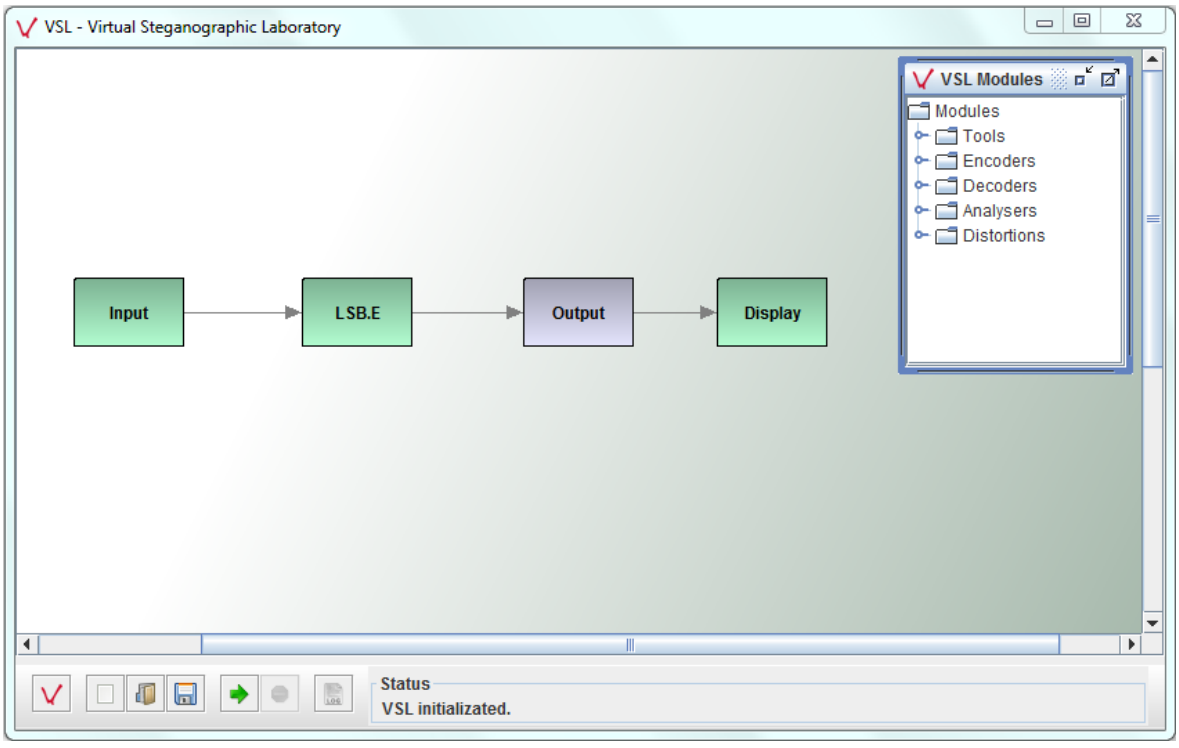
Mesaj metninin kaydedilmesi, "Extract & Save As Text" butonuna basıldığında başlayan veri ayıklama aşamasının son adımıdır. Bir önceki adımda elde edilen *outtext* string değişkeni, *System.StreamWriter* sınıfının *Write()* metodu sayesinde metin dosyasına yazılır.

Bir dosya kayıt ekranı (SaveFileDialog) açılarak, kullanıcıya, mesaj metin dosyasının kaydedileceği dizin seçtirilir. Metin dosyasına isim verilerek "Kaydet" (veya "Save") tuşuna basıldığında, uygulamanın veri ayıklama modülü tamamlanmış olur. Bu durumda "*Text File Path*" yazısı altındaki metin kutusunda, kaydedilen mesaj metni dosyasının tam yolu görülecektir. Ayrıca etiket metni de "Status: Extraction completed… Ready" olarak görünecektir.

5. K5 UYGULAMASININ BULGULARI İLE DİĞER STEGANOGRAFİ UYGULAMALARININ BULGULARININ KARŞILAŞTIRILMASI

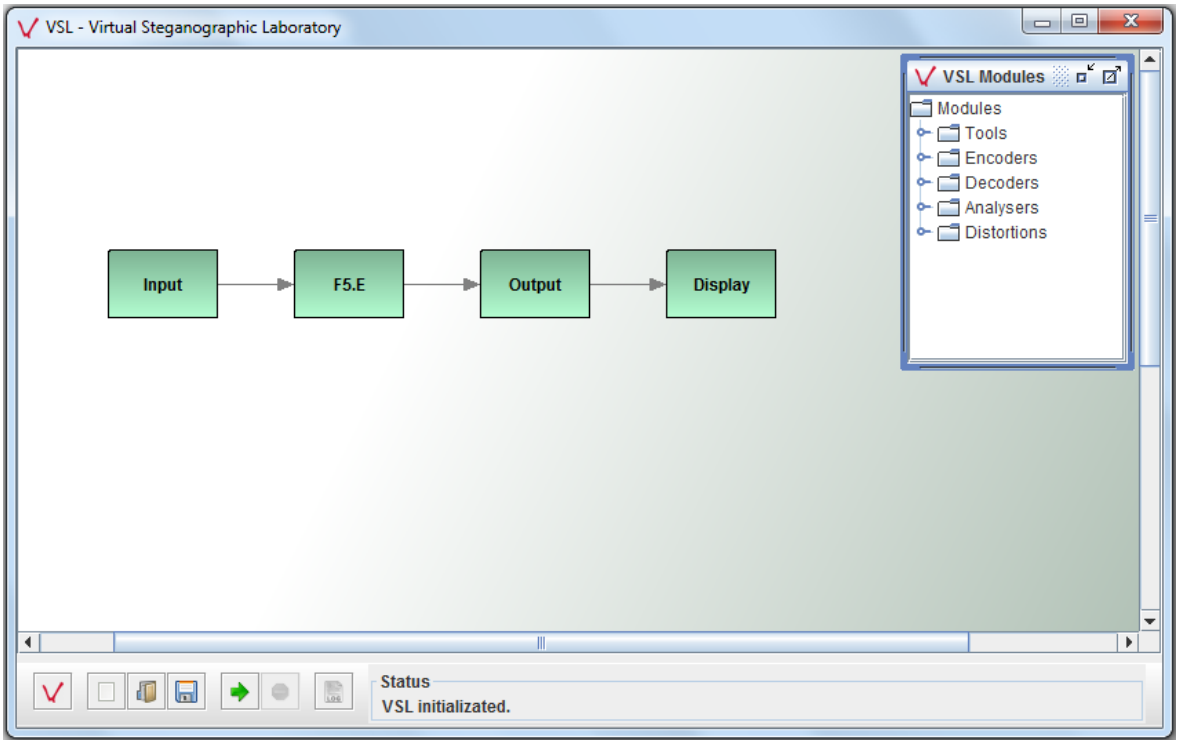
Bir önceki bölümde işleyişi anlatılan K5 Steganografi uygulaması, piksellerin renk değerlerinin son bitlerinde değişiklik yaparak veri gömme işlemini gerçekleştirmesi bakımından, LSB algoritmasında olduğu gibi, uzamsal tabanlı teknikle çalışmaktadır. Bunun yanı sıra, sıralamalı ayrım ve matris kodlama işlemlerini içermesi dolayısıyla, dönüşüm tabanlı çalışan F5 algoritması ile de benzerlik göstermektedir.

Bu bölümde, önerilen K5 Steganografi uygulaması ile bahsedilen diğer iki algoritmanın kullanıldığı uygulamalar karşılaştırılmıştır. Karşılaştırma için ki-kare testi içeren bir steganaliz saldırısı düzenlemeye imkan tanıyan "Ki-Kare Steganalizi (Chi-Square Steganalysis)" uygulaması kodlanmıştır. Ki-kare testinin seçilme sebebi ise, uzamsal tabanlı çalışan steganografi algoritmalarında fark edilemezlik ölçümü için kullanımının elverişli olmasındandır [38].



Resim 5.1. VSL Tool yazılımının LSB.E ekranı

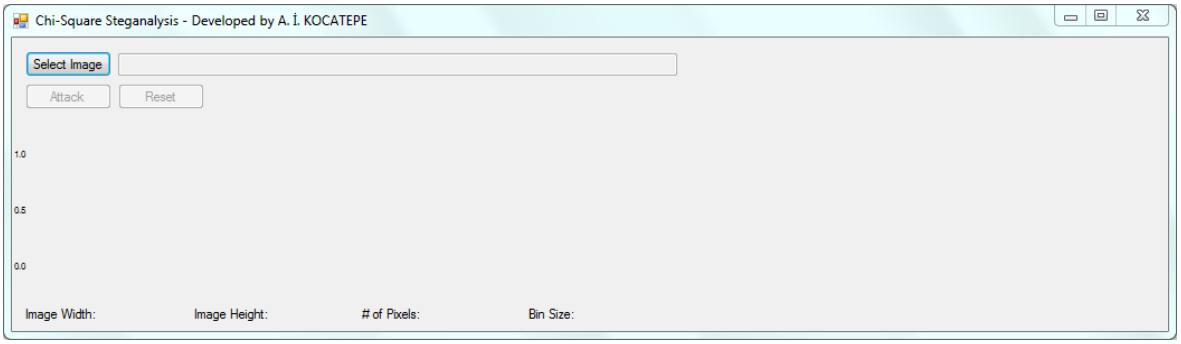
LSB ve F5 algoritmaları kullanılarak veri gömme ve ayıklama işlemleri, VSL Tool yazılımı ile gerçekleştirilmiştir [55]. Resim 5.1'de bu yazılımın LSB algoritması kullanılan modülünün ekran görüntüsü mevcuttur. Resim 5.2'de ise VSL Tool yazılımının F5 algoritması kullanılarak veri gömme işlemi gerçekleştiren modülünün ekran görüntüsü bulunmaktadır.



Resim 5.2. VSL Tool yazılımının F5.E ekranı

Yazılım ekranında bulunan "Input" kutusu, veri gömme sırasında kullanılacak olan taşıyıcı resmi seçmeyi sağlamaktadır. Kullanılan algoritmanın bulunduğu kutu ile ise hem gizlenecek veri dosyası seçilebilmekte hem de algoritmanın parametreleri (F5'in şifresi gibi) ayarlanabilmektedir. "Output" isimli kutu sayesinde, veri gömme işlemi sonucu elde edilecek olan stego-resmin nereye ve hangi isim/uzantı ile kaydedileceği seçilebilmektedir. Buna rağmen, F5 algoritması kullanıldığı durumda Output kutusundan dosya uzantısı ne seçilirse seçilsin, üretilen stego-resim JPEG biçiminde olmaktadır. Son olarak "Display" kutusu sayesinde ise uygulamanın çıktıları görüntülenebilmektedir.

LSB, F5 ve K5 kullanılarak elde edilen stego-resimlerin steganalizi için "Ki-Kare Steganalizi (Chi-Square Steganalysis)" uygulaması kodlanmıştır. Bu uygulamanın ana ekran görüntüsü Resim 5.3'te görülmektedir.



Resim 5.3. Ki-Kare Steganalizi uygulamasının ana ekran görüntüsü

Uygulama ekranındaki "Select Image" butonu, saldırı düzenlenecek olan stego-resmin seçilmesini sağlamaktadır. K5 uygulamasındaki dosya seçme butonları gibi "Select Image" butonuna tıklandığında da bir dosya açma ekranı (OpenFileDialog) gelmektedir. Bu ekrandaki filtreler de yine sadece resim dosyalarının (*.bmp, *.png ve *.jpg) seçilmesine izin vermektedir.

Stego-resim seçildiğinde dosyanın tam yolu metin kutusunda görünmektedir. Bununla birlikte seçilen stego-resmin en ve boy değerleri *w* ve *h* isimli tamsayı değişkenlere aktarılmaktadır. Burada *w* * *h* çarpımının dosyadaki piksel sayısını vereceği dikkate alınarak, ki-kare testinde kullanılacak ölçek boyutu olan *binsize* değişkeni Eşitlik 5.1'deki gibi elde edilmektedir.

$$binsize = (w * h) / 1000 \tag{5.1}$$

Uygulama ekranının alt kısmındaki etiketlere *w*, *h*, piksel sayısı ve *binsize* yazdırılmaktadır. Ayrıca "Attack" butonu da etkinleştirilerek ki-kare saldırısı başlatılmaya hazır hale getirilmektedir.

Seçilen stego-resim üzerine düzenlenecek olan ki-kare saldırısının işlemleri, "Attack" butonuna basılmasıyla başlar. Öncelikle, stego-resmin tüm pikselleri üzerinde işlem yapılmasını sağlayacak olan dış döngü başlatılır. Bu dış döngünün kontrol şartı her adımda *binsize* kadar artacak olan bir değişken ile sağlanmakta ve üst sınır olarak piksel sayısı (*w* * *h*) dikkate alınmaktadır.

İç döngüde ise *binsize* adedinde piksel üzerinde işlem yapılmaktadır. Her bir pikselin üç renk değerleri, önceden sıfırlanmış 1x256 boyutunda üç farklı dizi (*redf, grnf* ve *bluf*) kullanılarak toplamalı bir şekilde sayılır. Böylece her üç dizideki elemanların, indislerinin belirttiği renk değerlerinin sayısını taşımaları sağlanmış olur.

Örneğin; bir *binsize* içerisinde kırmızı renk değeri 55 olan 10 piksel varsa, kırmızı rengi kaydeden dizinin 55 indisli elemanının değeri 10 olacaktır (*redf*[55] = 10). Bir *binsize* içindeki tüm pikseller için bu renk sayısı değerleri bulunduktan sonra, dizilerin her elemanı *binsize* değerine bölünerek, renk değerlerinin frekansları elde edilmiş olur. Bu frekanslar, ki-kare testindeki gözlenen (o_i) değerlerdir.

Beklenen değerleri (e_i) elde etmek için ise yine her bir renk için 1x256 boyutunda üç farklı dizi (*redE, grnE* ve *bluE*) kullanılır. Bölüm 3.3.1'de anlatıldığı gibi beklenen frekans değerleri, ardışık değer çiftlerinin (pairs of values) frekans değerlerinin ortalamasına eşit olacaktır. Bu hesaplamalar Eşitlik 5.2'de gösterilmektedir.

$$xE[2*i] = xE[2*i+1] = \frac{xf[2*i] + xf[2*i+1]}{2} \; ; \; 0 \leq i < 128 \; ; \; x \in \{blu, grn, red\} \qquad (5.2)$$

Her bir renk için gözlenen ve beklenen frekans değerleri yanı sıra ki-kare toplamları ve serbestlik dereceleri (degree of freedom) de hesaplanarak, gama fonksiyonu kullanılması sayesinde ki-kare dağılımlı olasılık yoğunluk fonksiyonunun p değeri elde edilir.

Dış döngünün her *binsize*'lık adımında elde edilen 1 – p değerinin bağıl değişimi, *CreateGraphics().DrawLine* metodu kullanılarak ekranda ayrılmış kısma çizdirilir. Böylece "Attack" butonunun işlevi sonlanmış olur ve "Reset" butonu aktive edilir.

Uygulamanın her test için tekrar kapatılıp açılmasının önüne geçerek ki-kare saldırı işlemini kolaylaştırma amacıyla tasarlanan "Reset" butonuna tıklandığında; ekrana çizilmiş olan grafikler silinir, dosya yolunu gösteren metin kutusu temizlenir ve "Select Image" butonu aktive edilerek yeni bir steganalize imkan tanınmış olur.

5.1. Ki-kare Saldırısında Kullanılan Dosyalar

Önerilen K5 Steganografi uygulaması ile LSB ve F5 algoritmalarının karşılaştırılması amacıyla; Resim 5.4'te ölçeklendirilmiş olarak görülen JPEG, PNG ve BMP biçimlerindeki üç resim dosyası taşıyıcı nesne olarak kullanılmıştır.

Resim 5.4. Taşıyıcı resimler a) 01.lena.jpg b) 02.swirl.png c) 03.paint.bmp

Bu taşıyıcı resimlere ait bazı özellikler ise Çizelge 5.1'de verilmiştir. Bu özellikler, ki-kare saldırılarının karşılaştırılmasında kullanılmaktadır.

Çizelge 5.1. Taşıyıcı resimlere ait özellikler

Dosya adı	**Genişlik (*w*)**	**Yükseklik (*h*)**	**Piksel sayısı**	**Dosya boyutu (bayt)**
01.lena.jpg	740	729	539460	111458
02.swirl.png	200	200	40000	25293
03.paint.bmp	658	475	312550	938654

Devam eden bölümlerde görüleceği üzere LSB, F5 ve K5'i karşılaştırmak için üç kategoride birbirine denk steganografik işlemler gerçekleştirilebilecek durumlar hazırlanmıştır. Bu durumlarda yük (payload) olarak kullanılmaya uygun hazırlanan metin (*.txt) dosyalarına ait bilgiler ise Çizelge 5.2'de verilmiştir. Çizelgede adları verilen dosyalar düz metinden (plain text) oluştukları için bulundurdukları karakter sayıları, sahip oldukları dosya boyutuna eşittir.

Çizelge 5.2. Yük olarak kullanılan metin dosyaları

Dosya adı	**Karakter sayısı / Dosya boyutu (bayt)**
01a.cover_length.txt	107892
01b.cover_size_0.1.txt	11146
01c.cover_size_0.5.txt	55729
02a.cover_length.txt	8000
02b.cover_size_0.1.txt	2529
02c.cover_size_0.5.txt	12646
03a.cover_length.txt	62510
03b.cover_size_0.1.txt	93865
03c.cover_size_0.5.txt	469327

5.2. Uygulamaların Karşılaştırılması için Tasarlanan Testler

Belirtilen taşıyıcı resimler ve oluşturulan yükler sayesinde A, B ve C olarak adlandırılan üç farklı test düzeneği oluşturulmuştur. Her üç testte de üç algoritmanın, üç farklı taşıyıcı resim ile çalıştırılması sonucunda toplam 27 adet stego-resim elde edilmiştir. Daha sonra Ki-kare Steganaliz uygulaması kullanılarak bu stego-resimler ki-kare saldırısına tabi tutulmuştur.

Düzenlenen ilk test olan Test-A'da, yük metni uzunluğunun (karakter sayısı) taşıyıcı resmin piksel sayısına oranı %20 değerinde sabit tutulmuştur. Sabit oranı yakalamak için JPEG taşıyıcıya "01a.cover_length.txt" isimli yük, PNG taşıyıcıya "02a.cover_length.txt" isimli yük ve BMP taşıyıcıya da "03a.cover_length.txt" isimli yük gömülmüştür. Veri gömme işlemleri her üç uygulama ile de gerçekleştirilip dokuz farklı stego-resim elde edilmiştir. LSB, F5 ve K5 uygulamaları ile elde edilen stego-resimlerin ki-kare saldırısı karşısında verdikleri ki-kare dağılımlı olasılık yoğunluk fonksiyonunun 1 - p değerleri gözlemlenmiş ve *binsize* artışına bağlı değişen grafikleri çizdirilmiştir.

Test-B'de ise yük olarak seçilen metin dosyası boyutunun (bayt) taşıyıcı resmin boyutuna (bayt) oranı %10 değerinde sabit tutulmuştur. Bu oranını ayarlamak için JPEG taşıyıcıya "01b.cover_size_0.1.txt" isimli yük, PNG taşıyıcıya "02b.cover_size_0.1.txt" isimli yük ve

BMP taşıyıcıya da “03b.cover_size_0.1.txt” isimli yük gömülmüştür. Veri gömme işlemleri LSB, F5 ve K5 uygulamaları ile gerçekleştirilip dokuz farklı stego-resim elde edilmiş ve bu stego-resimlere ki-kare saldırısı uygulanmıştır. Üç farklı uygulama sonucunda elde edilen stego-resimlerin ki-kare saldırısı karşısında verdikleri 1 - p değerleri gözlemlenerek *binsize* artışına göre değişimleri grafikleştirilmiştir.

Son olarak Test-C ile yük boyutu, taşıyıcı resmin boyutunun yarısı olacak şekilde ayarlanmıştır. Bu oranı sağlamak için JPEG taşıyıcıya “01c.cover_size_0.5.txt” isimli yük, PNG taşıyıcıya “02c.cover_size_0.5.txt” isimli yük ve BMP taşıyıcıya da “03c.cover_size_0.5.txt” isimli yük gömülmüştür. LSB, F5 ve K5 uygulamaları ile elde edilen dokuz farklı stego-resim için Ki-kare Steganaliz uygulaması sayesinde 1 – p değerlerinin grafikleri çizdirilmiştir.

Testler sonucu elde edilen grafiklerin karşılaştırılması JPEG, PNG ve BMP taşıyıcı dosyaları için ayrı ayrı yapılmıştır. Her karşılaştırmalı grafikte LSB, F5 ve K5 uygulamalarının ürettikleri stego-resimlerin ki-kare test sonuçları bir arada ele alınmıştır. Sonuç bölümünde bu grafiklere yer verilerek yorumlar eklenmiştir.

6. SONUÇ

Bu tez çalışması, sıralamalı ayrım ve matris kodlama işlemlerini içeren ve aynı zamanda çok biçimli resim destekleyen bir steganografi uygulaması geliştirme amacı ile başlamıştır. Bir steganogramın fark edilemezlik katmanını aşmak için uygulanan steganaliz saldırıları söz konusu olduğunda, ki-kare testi gibi istatistiksel yöntemlerin etkili sonuç verdiği yapılan araştırmalar neticesinde belirlenmiştir [38]. Bununla birlikte, ki-kare testine karşı da sıralamalı ayrım ve matris kodlama gibi yöntemlerin kullanılarak fark edilemezliğin korunabildiği ortaya konmuştur [33]. Ancak bu özelliklerin dönüşüm tabanlı steganografi teknikleriyle birlikte kullanımı, yalnızca belli biçime sahip resim dosyalarıyla işlem yapabilme kısıtını beraberinde getirmektedir.

Bu kısıtı taşımakla birlikte, özellikle ki-kare saldırısına karşı fark edilemezliği yüksek olması ve sıralamalı ayrım ile matris kodlama kullanılması sebebiyle F5 algoritması incelenerek çalışma prensipleri ayrıntılı bir şekilde çözümlenmiştir. F5'in de yine bir başka steganografi algoritması olan Jsteg üzerine iyileştirmeler yapılarak geliştirildiği öğrenilmiştir [33]. Bunun üzerine, tez çalışması sonucunda geliştirilmesi planlanan uygulama için F5 algoritmasının model alınmasına ve daha detaylı incelenmesine karar verilmiştir.

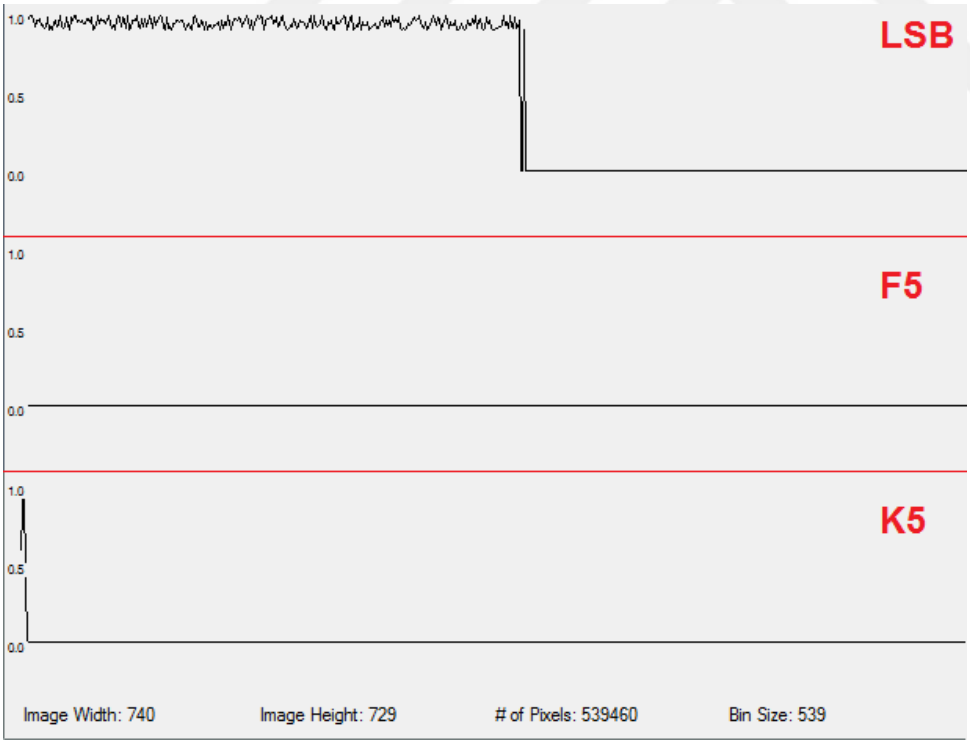
Belirlenen amaç doğrultusunda, F5 algoritmasının ki-kare testine karşı gösterdiği performansa sahip olmakla birlikte, dönüşüm tabanlı tekniklerden kaynaklanan kısıtlı dosya biçimi desteklenmesi sorununa çözüm olabilecek bir uygulama olarak K5 Steganografi uygulaması tasarlanmıştır. Tasarımın gerçekleştirilebileceği uygun yöntem ve araçlar ile alandaki diğer çalışmalar incelenerek, sıralamalı ayrım ve matris kodlama işlemlerinin uzamsal tabanda çalıştırılması sayesinde bahsedilen dosya biçimi kısıtının aşılabileceği kanaatine varılmıştır.

Önerilen K5 Steganografi uygulamasının bahsedilen kısıtı ortadan kaldırmakla birlikte, fark edilemezlik ölçümü için ki-kare saldırısında karşılaştırılabilmesi hedefiyle F5'in yanı sıra uzamsal tabanlı teknikle çalışması sebebiyle LSB algoritması da seçilmiştir. Böylece geliştirilen K5 Steganografi uygulamasının belirlenen tez amacına uygun olup olmadığını saptamaya yönelik çalışmalar gerçekleştirilmiştir.

Tasarlanan üç farklı test sayesinde LSB, F5 ve K5 uygulamalarının farklı taşıyıcı resim ve yük dosyası şartları altında ki-kare testine karşı gösterdikleri tepkiler elde edilerek karşılaştırılmıştır. Yapılan steganalizler sonucu edinilen bulgular, ki-kare dağılımlı olasılık yoğunluk fonksiyonunun 1 – p değerini grafiksel olarak ortaya koymaktadır. Testler sonucu oluşan grafikler, taşıyıcı resim bazında üç farklı algoritma için karşılaştırmalı olarak sunulmuştur.

6.1. Test Sonuçlarının Karşılaştırmalı Analizi

Test-A'da, yük metni uzunluğunun (karakter sayısı) taşıyıcı resmin piksel sayısına oranı %20 değerinde sabit tutulmuştur. LSB, F5 ve K5 uygulamaları ile veri gömülerek elde edilen stego-resimlerin ki-kare saldırısı sonucunda çizdirilen 1 - p grafikleri karşılaştırmalı olarak incelenmiştir. Şekil 6.1'de JPEG taşıyıcı kullanılan durumda üç farklı algoritmanın ki-kare sonuçları görülmektedir.



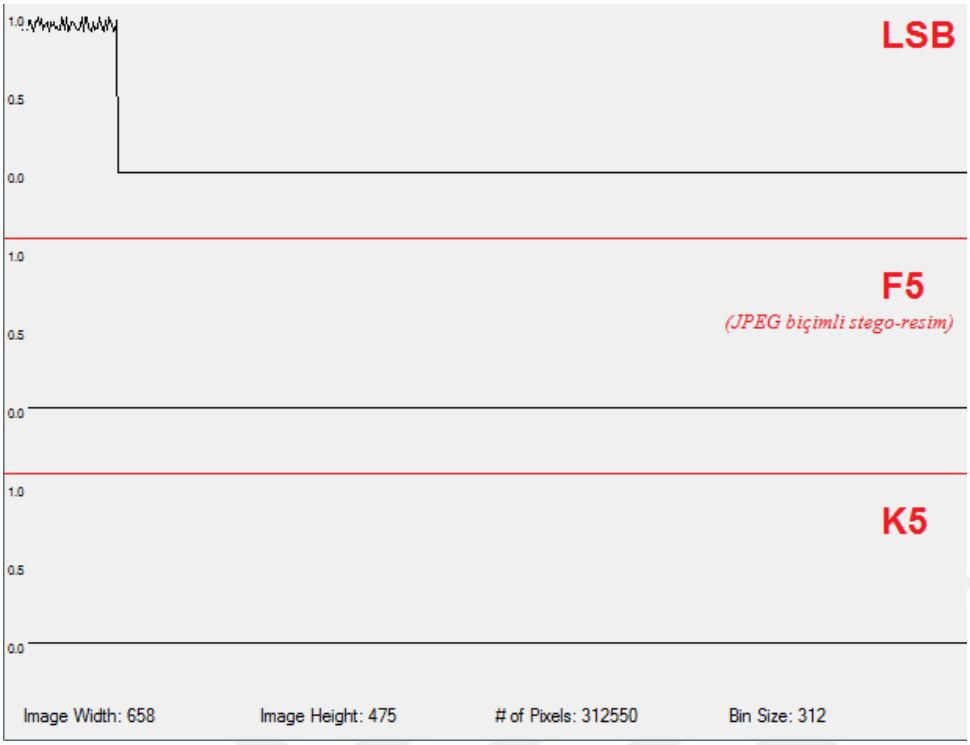
Şekil 6.1. JPEG taşıyıcı ve "01a.cover_length.txt" isimli yük durumunun steganalizi

Grafik çizimi sonucunda düşey eksende (1 – p) 1.0 değerine yakın bölgeler bulunması, kullanılan taşıyıcı resimde gizlenmiş veri olma ihtimalinin yüksekliğini belirtmektedir [38]. Ayrıca grafiğin düşeyde yaptığı ani değişimler de veri gömme işleminin sonlandığı bölümleri ortaya çıkarır. Resim piksellerine veri gizlerken düzgün sıra takip eden bir steganografi uygulamasının grafiğinde, yatay eksendeki eşdeğer ilerlemelere karşın düşey eksende keskin değer farkları bulunması beklenir.

Test-A'nın PNG biçimli taşıyıcı resim ile gerçekleştirilmesi sonucu üç farklı algoritmanın ki-kare saldırısına ait karşılaştırmalı grafikler Şekil 6.2'de gösterilmiştir. Testin bu aşamasında LSB ve K5 uygulamalarından PNG biçimli, F5'ten ise mecburen JPEG biçimli stego-resim elde edildiği dikkate alınmalıdır.

Şekil 6.2. PNG taşıyıcı ve "02a.cover_length.txt" isimli yük durumunun steganalizi

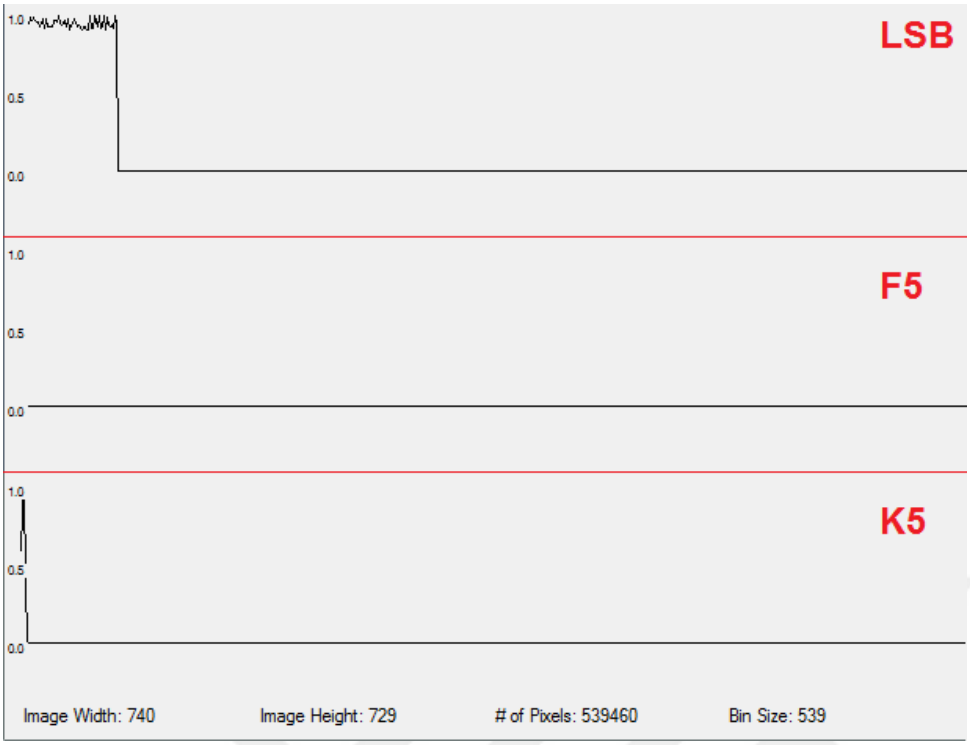
JPEG ve PNG biçimli taşıyıcı resimlerde uygulanan aynı şartların sağlandığı Test-A'nın BMP taşıyıcı resmi kullanılmasıyla elde edilen stego-resimlerinin ki-kare test sonuç grafikleri ise Şekil 6.3'te görülmektedir. Burada da F5 uygulaması sonucu ancak JPEG biçimli stego-resim oluşurulabildiği not edilmiştir.



Şekil 6.3. BMP taşıyıcı ve "03a.cover_length.txt" isimli yük durumunun steganalizi

İkinci karşılaştırma durumlarını içeren Test-B'de ise, yük olarak seçilen metin dosyası boyutunun (bayt) taşıyıcı resmin boyutuna (bayt) oranı %10 değerinde sabit tutulmuştur. Veri gömme işlemleri LSB, F5 ve K5 uygulamaları ile gerçekleştirilip dokuz farklı stego-resim elde edilmiş ve bu stego-resimlere ki-kare saldırısı uygulanmıştır. Böylece, üretilen stego-resimlerin ki-kare saldırısı karşısında verdikleri 1 - p değerleri gözlemlenerek karşılaştırmalı grafikler oluşturulmuştur.

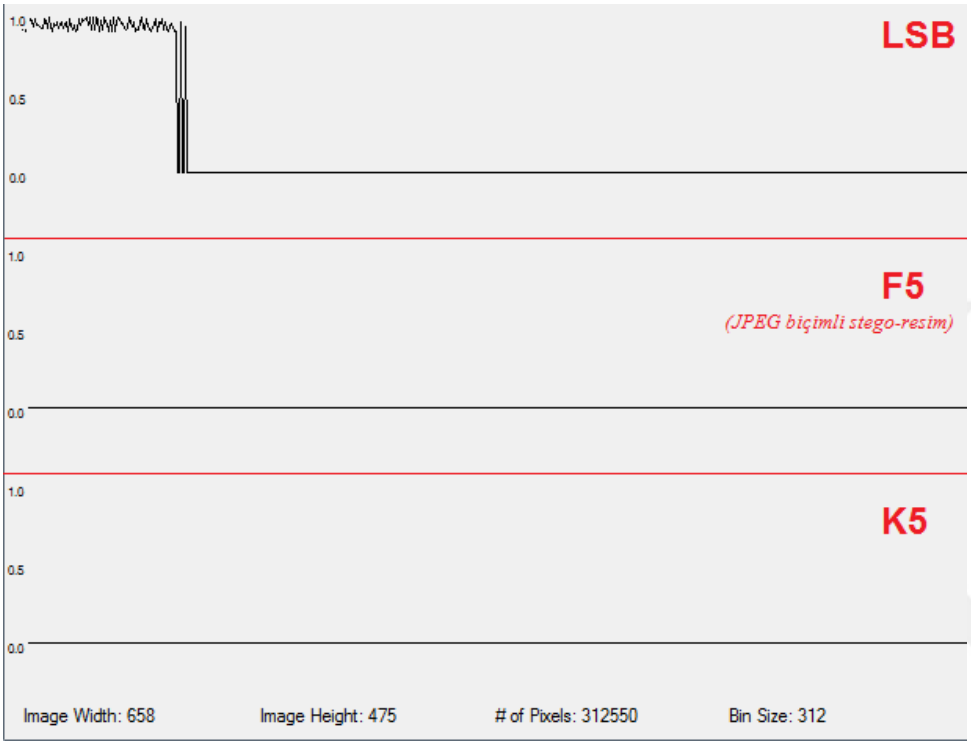
Şekil 6.4'te JPEG taşıyıcı kullanılan durumda üç farklı algoritmanın ki-kare saldırısı sonuçları görülmektedir. Test-B'de PNG biçimli taşıyıcı resim kullanıldığı durumda LSB, F5 ve K5 uygulamaları ile elde edilen stego-resimlerin ki-kare test grafiklerinin karşılaştırmalı görüntüsü ise Şekil 6.5'te verilmiş ve F5 uygulamasının ürettiği JPEG biçimli stego-resim belirtilmiştir.



Şekil 6.4. JPEG taşıyıcı ve "01b.cover_size_0.1.txt" isimli yük durumunun steganalizi

Şekil 6.5. PNG taşıyıcı ve "02b.cover_size_0.1.txt" isimli yük durumunun steganalizi

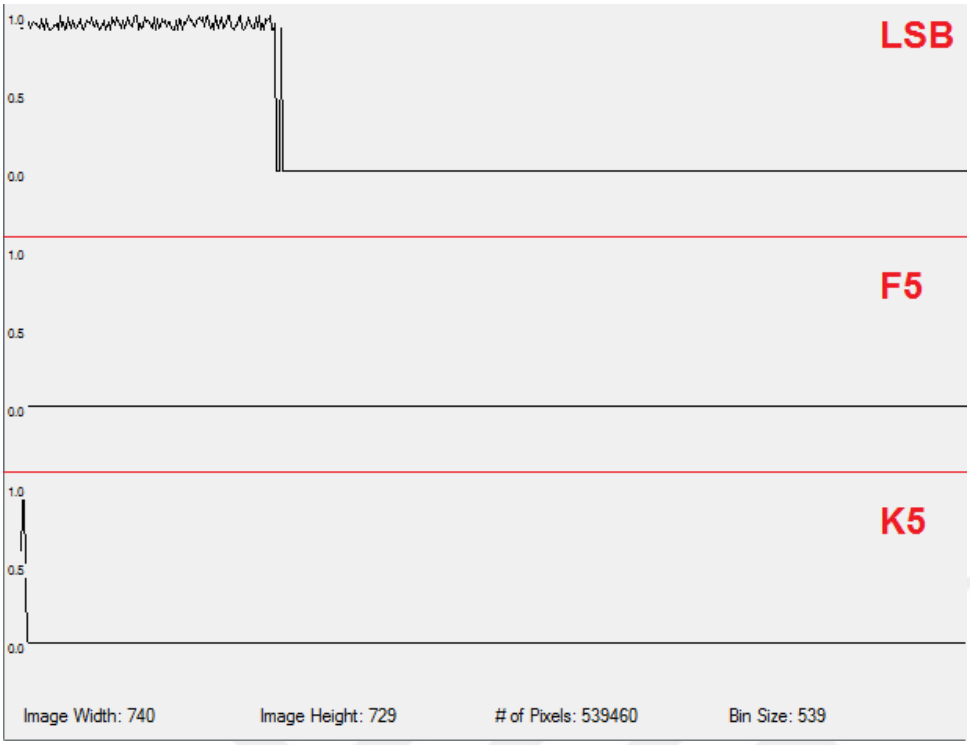
Aynı boyut oranı (%10) kullanılarak Test-B'nin BMP biçimli taşıyıcı resim için üç farklı steganografi uygulamasıyla üretilmiş stego-resimlere ki-kare saldırısı gerçekleştirilmiş ve Şekil 6.6'da görülen karşılaştırmalı grafikler elde edilmiştir. F5 uygulamasının değişmez çıktısı olan JPEG biçimli stego-resim not edilmiştir.



Şekil 6.6. BMP taşıyıcı ve "03b.cover_size_0.1.txt" isimli yük durumunun steganalizi

Yük boyutunun taşıyıcı resmin boyutuna %50 oranlı olması durumu ele alınan Test-C'de; LSB, F5 ve K5 uygulamaları ile elde edilen dokuz farklı stego-resim için Ki-kare Steganaliz uygulaması sayesinde 1 – p değerlerinin grafikleri çizdirilmiştir. JPEG biçimli taşıyıcı resmin kullanıldığı durumlar için karşılaştırmalı grafikler Şekil 6.7'de verilmiştir.

Şekil 6.8'de PNG biçimli taşıyıcı resim kullanılarak üretilen stego-resimlerin ki-kare saldırı grafikleri gösterilmiştir. LSB ve K5 uygulamaları ile PNG biçimli stego-resim elde edilebildiği halde, F5 sonucunda JPEG biçimli stego-resim oluştuğuna dikkat çekilmiştir.



Şekil 6.7. JPEG taşıyıcı ve "01c.cover_size_0.5.txt" isimli yük durumunun steganalizi

Şekil 6.8. PNG taşıyıcı ve "02c.cover_size_0.5.txt" isimli yük durumunun steganalizi

BMP biçimli taşıyıcı resim kullanılarak üretilen stego-resimlere Ki-Kare Steganaliz uygulaması sayesinde gerçekleştirilen saldırılar sonucunda Şekil 6.9'daki grafikler çizdirilmiştir. F5 uygulaması ile zorunlu olarak JPEG biçimli stego-resim oluştuğu not edilmiştir.

Şekil 6.9. BMP taşıyıcı ve "03c.cover_size_0.5.txt" isimli yük durumunun steganalizi

Grafiklerin karşılaştırmalı incelenmesi sonucunda ki-kare saldırısına karşı herhangi bir özel işleme sahip olmayan LSB algoritmasının fark edilemezlik özelliğini diğer iki algoritmaya kıyasla çok çabuk kaybettiği açıkça anlaşılabilmektedir. Ayrıca Şekil 5.9 ve Şekil 5.1'deki grafikler dikkate alındığında, yük boyutunun taşıyıcı boyutuna olan oranındaki artıştan oldukça etkilendiği sonucuna ulaşılabilir.

Testler sonucu oluşan grafiklerde bir başka dikkat çeken özellik ise F5 algoritmasının, taşıyıcı ve yük durumları ne olursa olsun gerçekleştirilen ki-kare saldırılarında fark edilemezlik özelliğini koruyor olduğudur. Ancak daha önce de belirtildiği gibi, F5 kullanan uygulamalarda taşıyıcı resim biçiminden bağımsız olarak stego-resim sadece JPEG biçiminde elde edilebilmektedir.

Geliştirilen K5 Steganografi uygulamasının ki-kare sonuçları grafiklerden incelendiğinde, PNG ve BMP biçimli taşıyıcı resimler kullanıldığı durumlarda F5 ile aynı performansı ortaya koyduğu görülmektedir. Bir diğer dikkat çeken nokta ise taşıyıcı resmin JPEG olduğu durumlarda K5'in grafiğinin başlangıcında görülen kısa süreli ani yükseliş ve hemen ardından gelen keskin düşüştür. Şekil 6.1, 6.4 ve 6.7 incelenerek bu durum ayırdedilebilmektedir.

Yapılan araştırmalar sonucunda, ki-kare testi kullanılan diğer bazı çalışmalarda da grafikte küçük ölçekte beklenmeyen düşey hareketler gözlemlendiği bilgisine ulaşılmıştır [56]. Ayrıca ki-kare testinin kümülatif yapısından ötürü, belli bir sayıda örnekleme ulaşılmadığı durumlarda testin beklenen değer ve gözlenen değer arasındaki ilişkiyi saptamada gecikmeli sonuç verdiği anlaşılmıştır. Bu bağlamda, JPEG taşıyıcı kullanıldığı durumlarda K5'in grafiklerinde gözlemlenen sonucun fark edilemezliğin kaybı olarak yorumlanamayacağı kanısına ulaşılmıştır.

Ulaşılan tüm bulguların karşılaştırmalı analizi sonucunda; önerilen K5 Steganografi uygulamasının fark edilemezlik değerinde F5 ile aynı sayılabilecek performansa sahip olmakla birlikte, farklı biçimlerde stego-nesneler üretebilme kabiliyeti sayesinde resim steganografisi alanında yenilikçi olduğu sonucuna varılabilmektedir. Dolayısıyla bu tez çalışmasının amacı olan fark edilemezliği koruyarak çok biçimli resim destekleyen bir steganografi uygulaması elde edilmiştir.

KAYNAKLAR

1. Cheddad, A., Condell, J., Curran, K. ve Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal Processing*, 90(3), 727-752.

2. İnternet: Digital Watermarking Alliance. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.digitalwatermarkingalliance.org%2Fglossary.asp&date=2016-08-30, Son Erişim Tarihi: 30.08.2016.

3. Coşkun, İ., Akar, F. ve Çetin, Ö. (2013). A new digital image steganography algorithm based on visible wavelength. *Turkish Journal of Electrical Engineering & Computer Sciences*, 2013(21), 548-564.

4. Zielińska, E., Mazurczyk, W. ve Szczypiorski, K. (2012). Development Trends in Steganography. *Cornell University Library*, (1202.5289).

5. İnternet: Schulzrinne, H. (2008). Introduction to Cryptography. *Lecture Notes in CSW4180: Network Security*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.cs.columbia.edu%2F%7Ehgs%2Fteaching%2Fsecurity&date=2016-09-16, Son Erişim Tarihi: 16.09.2016.

6. Raggo, M. ve Hosmer, C. (2012). *Data Hiding.* Syngress: An Imprint of Elsevier, 1-67, 181-191, 229-236.

7. Subhedar, M. S. ve Mankar, V. H. (2014). Current status and key issues in image steganography: A survey. *Computer Science Review*, 2014(13-14), 95-113.

8. Li, B., He, J., Huang, J. ve Shi, Y. Q. (2011). A survey on image steganography and steganalysis. *Journal of Information Hiding and Multimedia Signal Processing*, 2(2), 142-172.

9. Cole, E. (2003). *Hiding in Plain Sight: Steganography and the Art of Covert Communication*. New Jersey: Wiley.

10. Wayner, P. (2009). *Disappearing Cryptography, Information Hiding: Steganography & Watermarking* (3rd edition). Massachusetts: Morgan Kaufmann.

11. Judge, J. C. (2001). Steganography: Past, Present, Future. *SANS Institute – InfoSec Reading Room,* 552-580.

12. Sencar, H. T., Ramkumar, M. ve Akansu, A. N. (2004). *Data Hiding – Fundamentals and Applications.* California: Elsevier Academic Press.

13. Simmons, G. J. (1984). The Prisoners' Problem and the Subliminal Channel. *Sandia National Laboratories,* 87185, 51-67.

14. İnternet: Petitcolas, F. History of steganography and cryptography. *The information hiding homepage*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.petitcolas.net%2Fsteganography%2Fhistory.html&date=2016-08-30, Son Erişim Tarihi: 30.08.2016.

15. Anderson, R. (1996). Stretching the Limits of Steganography. *Information Hiding*, 39-48.

16. İnternet: Researchers: No secret bin Laden messages on sites. (2001). *USA Today – Tech*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fusatoday30.usatoday.com%2Flife%2Fcyber%2Ftech%2F2001%2F10%2F17%2Fbin-laden-site.htm&date=2016-08-30, Son Erişim Tarihi: 30.08.2016.

17. Sağıroğlu, Ş. ve Tunçkanat, M. (2002). A Secure Internet Communication Tool. *Turkish Journal of Telecommunications,* 1(1), 40-46.

18. İnternet: Government uses color laser printer technology to track documents. (2004). *PCWorld*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.pcworld.com%2Farticle%2F118664%2Fgovernment_uses_color_laser_printer_technology_to_track_documents.html&date=2016-08-30, Son Erişim Tarihi: 30.08.2016.

19. Hosmer, C. (2013). *Steganography*. George Mason University – Cybersecurity Innovation Forum, Virginia, ABD.

20. Şahin, A., Buluş, E. ve Sakallı, M. T. (2006). *Gri Seviye Resimler Üzerinde Rastgele LSB Yöntemini ve Sayı Teorisini Kullanarak Bilgi Gizleme ve Steganaliz*, Akademik Bilişim Konferansları, Denizli.

21. Ulutaş, M., Nabiyev, V. V. ve Ulutaş, G. (2009). Improvements in Geometry-Based Secret Image Sharing Approach with Steganography. *Mathematical Problems in Engineering*.

22. Ünlü, O. (2012). *Ortam ve Yöntem Bağımsız Steganografik Kütüphane Tasarımı*. Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

23. Ünlü, O. ve Karacan, H. (2013). Lattice approach to video steganography. *Turkish Journal of Electrical Engineering & Computer Sciences,* 2015(23), 2074-2088.

24. Bansal, D. ve Chhikara, R. (2014). An improved DCT based steganography technique. *International Journal of Computer Applications,* 102(14),46-49.

25. Karakış, R. (2015). *Epileptik MRG ve EEG Verileri için Bulanık Mantık Tabanlı Steganografi Uygulaması*. Doktora Tezi, Gazi Üniversitesi Bilişim Enstitüsü, Ankara.

26. Durdu, A. ve Özcerit, A. T. (2015). Güvenli iletişim için yeni bir veri gizleme algoritması. *Uluslararası Bilgi Güvenliği Mühendisliği Dergisi,* 1(1), 32-36.

27. Lin, E. T. ve Delp, E. J. (1999). A Review of Data Hiding in Digital Images; CERIAS 2001-139. *CERIAS Tech Report*, Indiana, ABD.

28. Hassan, A. A. (2015). *New Approach for Text Based Steganography*. Yüksek Lisans Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya.

29. Bender, W., Gruhl, D., Morimoto, N. ve Lu, A. (1996). Techniques for data hiding. *IBM Systems Journal,* 35(3&4), 313-336.

30. Zhang, X. ve Wang, S. (2005). Steganography using multiple-base notational system and human vision sensitivity. *IEEE Signal Processing Letters,* 12(1), 67-70.

31. Wu, D. C. ve Tsai, W. H. (2003). A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters,* 24(9-10), 1613-1626.

32. Chen, B., ve Wornell, G. W. (2001). Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory,* 47(4), 1423-1443.

33. Westfeld, A. (2001). *F5 – A Steganographic Algorithm: High Capacity Despite Better Steganalysis*, Proceedings of 4th International Workshop on Information Hiding, LNCS 2137, Pennsylvania, ABD.

34. Marvel, L. M., Boncelet, Jr., C. G. ve Retter, C. T. (1998). Methodology of Spread-Spectrum Image Steganography; ARL TR-1698. *Army Research Laboratory Report*, Maryland, ABD.

35. Sallee, P. (2005). Model-based methods for steganography and steganalysis, *International Journal of Image and Graphics,* 05(01), 167-189.

36. Rowland, C. (1997). Covert channels in the TCP/IP protocol suite. *First Monday,* 2(5).

37. Chan, C. K. ve Cheng, L. M. (2004). Hiding data in images by simple LSB substitution, *Pattern Recognition*, 37:469-474.

38. Westfeld, A., Pfitzmann, A. (2000) Attacks on Steganographic Systems: Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools–and Some Lessons Learned, *Information Hiding*, 1768, 61-76.

39. Fridrich, J. ve Goljan, M. (2002). *Practical Steganalysis of Digital Images - State of the Art*. Proceedings of SPIE, 2002(4675), 1-13.

40. Harmsen, J. J. ve Pearlman, W. A. (2003). *Steganalysis of additive noise modelable information hiding,* Electronic Imaging Conference, California, ABD.

41. Kharrazi, M., Sencar, H. T. ve Memon, N. (2006). Performance study of common image steganography and steganalysis techniques. *Journal of Electronic Imaging,* 15(4).

42. Westfeld, A. (2007, 14 Haziran). *ROC Curves for Steganalysts.* Proceedings of the 3rd Wavila Challenge, Saint Malo, Fransa, 39-45.

43. Geetha, S., Sindhu, S. S. ve Kamaraj, N. (2009). Blind image steganalysis based on content independent statistical measures maximizing the specificity and sensitivity of the system. *Computers & Security*, 2009(28), 683-697.

44. Liu, Q., Sung, A. H., Qiao, M., Chen, Z. ve Ribeiro, B. (2010). An improved approach to steganalysis of JPEG images. *Information Sciences*, 2010(180), 1643-1655.

45. Sabeti, V. Samavi, S., Mahdavi, M. ve Shirani, S. (2010). Steganalysis and payload estimation of embedding in pixel differences using neural networks. *Pattern Recognition*, 43, 405-415.

46. Li, L., Sencar, H. T. ve Memon, N. (2013). *A Cost-Effective Decision Tree Based Approach to Steganalysis.* SPIE-IS&T Electronic Imaging, 2013(8665), 1-7.

47. Sabnis, S. K. ve Awale, R. N. (2016). Statistical steganalysis of high capacity image steganography with cryptography. *Procedia Computer Science,* 79(2016), 321-327.

48. Richer, P. (2003). Steganalysis: Detecting hidden information with computer forensic analysis. *SANS Institute – InfoSec Reading Room,* 1024-1036.

49. Kaur, M. ve Kaur, G. (2014). Review of various steganalysis techniques. *International Journal of Computer Science and Information Technologies*, 5(2), 1744-1747.

50. Pal, S. K., Saxena, P. K. ve Muttoo S. K. (2002). *A systematic approach to steganalysis of images*. Proceedings of the Pacific Rim Workshop on Digital Steganography, STEG'02, Kitakyushu, Japonya, 179-188.

51. Huang, J. (2000). *Statistics of Natural Images and Models*, Ph. D. Thesis, Brown University Division of Applied Mathematics, Rhode Island, ABD.

52. Zhang, T. ve Ping, X. (2003). A new approach to reliable detection of LSB steganography in natural images. *Signal Processing*, 83(10), 2085-2093.

53. İnternet: Üçdoğruk, Ş. (2008). Ki-Kare Testleri. *İstatistik Ders Notları-10*, URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.deu.edu.tr%2Fuserweb%2Fs.ucdogruk%2Fistatistik&date=2016-11-16, Son Erişim Tarihi: 16.11.2016.

54. İnternet: ALGLIB – Numerical Analysis Library. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.alglib.net%2F&date=2016-12-14, Son Erişim Tarihi: 14.12.2016.

55. İnternet: Węgrzyn, M. (2009). VSL – Virtual Steganographic Laboratory for Digital Images. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fvsl.sourceforge.net%2F&date=2016-12-14, Son Erişim Tarihi: 14.12.2016.

56. Stanley, C. A. (2005). *Pairs of Values and the Chi-squared Attack*, Master's Thesis, Iowa State University Department of Mathematics, Iowa, ABD.

EKLER

EK-1. K5 Steganografi uygulamasının veri gömme modülü için C# kaynak kodları

```
    private void buttonSelectCover_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofdCover = new OpenFileDialog();
        ofdCover.Filter = "Image Files (*.bmp, *.png, *.jpg) | *.bmp; *.png; *.jpg";
        ofdCover.InitialDirectory = @"C:\Users\Default\Desktop";
        if (ofdCover.ShowDialog() == DialogResult.OK)
        {
            textBoxCoverPath.Text = ofdCover.FileName.ToString();
            pictureBox1.ImageLocation = textBoxCoverPath.Text;
            label2.Text = "Status: Cover Image is selected";
            coverEtkin = true;
            if (coverEtkin & mesajEtkin & pswEmbedEtkin)
                buttonEmbed.Enabled = true;
        }
    }
    private void buttonSelectMessage_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofdMesaj = new OpenFileDialog();
        ofdMesaj.Filter = "Text Files (*.txt) | *.txt";
        ofdMesaj.InitialDirectory = @"C:\Users\Default\Desktop";
        if (ofdMesaj.ShowDialog() == DialogResult.OK)
        {
            textBoxMessagePath.Text = ofdMesaj.FileName.ToString();
            gizlimesaj = File.ReadAllText(textBoxMessagePath.Text,
Encoding.Default);
            label2.Text = "Status: Message file is selected";
            mesajEtkin = true;
            if (coverEtkin & mesajEtkin & pswEmbedEtkin)
                buttonEmbed.Enabled = true;
        }
    }
    private void buttonPswEmbed_Click(object sender, EventArgs e)
    {
        while (true)
        {
            pswEmbed = Microsoft.VisualBasic.Interaction.InputBox("Password must be
at least 2 characters!",
                "Enter Password for Embedding");
            if (pswEmbed.Length >= 2) break;
        }
        label2.Text = "Status: Embedding password OK";
        pswEmbedEtkin = true;
        if (coverEtkin & mesajEtkin & pswEmbedEtkin)
            buttonEmbed.Enabled = true;
    }
    private void buttonEmbed_Click(object sender, EventArgs e)
    {
        buttonEmbed.Enabled = false;
        Bitmap img = new Bitmap(textBoxCoverPath.Text);
        int[] iarray, jarray;
        permStrad(img.Width, img.Height, pswEmbed, out iarray, out jarray);
        int boy = gizlimesaj.Length, say = boy; bool yaz = false;
        for (int i = 0; i < iarray.Length; i++)
            for (int j = 0; j < jarray.Length; j++)
                if (!yaz)
                    for (int k = 0; k < 4; k++)
                        if (boy >= 255)
                        {
        for (int i = 0; i < iarray.Length; i++)
```

EK-1 (devam). K5 Steganografi uygulamasının veri gömme modülü için C# kaynak kodları

```
            for (int j = 0; j < jarray.Length; j++)
                if (!yaz)
                    for (int k = 0; k < 4; k++)
                        if (boy >= 255)
                        {
                            Color p = img.GetPixel(iarray[i], jarray[j] * 4 + k);
                            img.SetPixel(iarray[i], jarray[j] * 4 + k,
Color.FromArgb(p.R, p.G, 255));
                            boy -= 255;
                        }
                        else if (boy > 0 & boy < 255)
                        {
                            Color p = img.GetPixel(iarray[i], jarray[j] * 4 + k);
                            img.SetPixel(iarray[i], jarray[j] * 4 + k,
Color.FromArgb(p.R, p.G, boy));
                            boy -= 255;
                        }
                        else
                        {
                            Color p = img.GetPixel(iarray[i], jarray[j] * 4 + k);
                            img.SetPixel(iarray[i], jarray[j] * 4 + k,
Color.FromArgb(p.R, p.G, 0));
                            yaz = true; k = 3;
                        }
                else if (yaz && say != 0)
                {
                    char harf =
Convert.ToChar(gizlimesaj.Substring(gizlimesaj.Length - say, 1));
                    string hrfbts = Convert.ToString(harf, 2).PadLeft(8, '0');
                    for (int k = 0; k < 4; k++)
                    {
                        Color p = img.GetPixel(iarray[i], jarray[j] * 4 + k);
                        char c1 = hrfbts[2 * k]; char c2 = hrfbts[2 * k + 1];
                        byte rnew, gnew, bnew;
                        matrixEncode(c1, c2, p, out rnew, out gnew, out bnew);

                        img.SetPixel(iarray[i], jarray[j] * 4 + k,
Color.FromArgb(rnew, gnew, bnew));
                    }
                    say--;
                }
                else{ j = jarray.Length - 1; i = iarray.Length - 1; }
        SaveFileDialog sfdImage = new SaveFileDialog();
        sfdImage.Filter = "Image Files (*.bmp, *.png, *.jpg) | *.bmp; *.png; *.jpg";
        sfdImage.InitialDirectory = @"C:\Users\Default\Desktop";
        if (sfdImage.ShowDialog() == DialogResult.OK)
        {
            textBoxCoverPath.Text = sfdImage.FileName.ToString();
            pictureBox1.ImageLocation = textBoxCoverPath.Text;
            img.Save(textBoxCoverPath.Text);
            label2.Text = "Status: Embedding completed... Ready";
            mesajEtkin = false; coverEtkin = false; pswEmbedEtkin = false;
            }
        }
```

EK-2. Sıralamalı ayrım, LFSR ve matris kodlama metotlarının C# kaynak kodları

```
    public void permStrad(int w, int h, string psw, out int[] iarray, out int[]
jarray)
    {
        List<int> ilist = new List<int>();
        List<int> jlist = new List<int>();
        string sw, sh; int[] swa, sha;
        char p0 = Convert.ToChar(psw.Substring(0, 1));
        string bits0 = Convert.ToString(p0, 2).PadLeft(8, '0');
        char p1 = Convert.ToChar(psw.Substring(1, 1));
        string bits1 = Convert.ToString(p1, 2).PadLeft(8, '0');
        int nw = (int)(Math.Truncate(Math.Log(w, 2)) + 1);
        int nh = (int)(Math.Truncate(Math.Log(h / 4, 2)) + 1);

        if (nw <= 8) sw = bits0.Substring(0, nw - 1) + '1';
        else sw = bits0.PadRight(nw, '1');
        lfsr(sw, out swa);

        if (nh <= 8) sh = bits1.Substring(0, nh - 1) + '1';
        else sh = bits1.PadRight(nh, '1');
        lfsr(sh, out sha);

        for (int i = 0; i < swa.Length; i++)
            if (swa[i] < w) ilist.Add(swa[i]);
        ilist.Add(0);
        iarray = ilist.ToArray(); ilist.Clear();

        for (int i = 0; i < sha.Length; i++)
            if (sha[i] < h / 4) jlist.Add(sha[i]);
        jlist.Add(0);
        jarray = jlist.ToArray(); jlist.Clear();
    }
    public void lfsr(string x, out int[] xarray)
    {
        List<int> xlist = new List<int>();
        int run = (int)(Math.Pow(2, x.Length) - 1);
        for (int i = 0; i < run; i++)
        {
            xlist.Add(Convert.ToInt32(x, 2));
            string srx = x.Substring(0, x.Length - 1);
            int xint = Convert.ToInt32(x, 2);
            int srxint = Convert.ToInt32(srx, 2);
            int xor = (xint ^ srxint);
            string lsb = Convert.ToString(xor, 2).PadLeft(x.Length, '0');
            lsb = lsb.Substring(lsb.Length - 1, 1);
            x = lsb + srx;
        }
        xarray = xlist.ToArray(); xlist.Clear();
    }
```

EK-2 (devam). Sıralamalı ayrım, LFSR ve matris kodlama metotlarının C# kaynak kodları

```
    public void matrixEncode(char m1, char m2, Color p,
        out byte ro, out byte go, out byte bo)
    {
        string r = Convert.ToString(p.R, 2).PadLeft(8, '0');
        string g = Convert.ToString(p.G, 2).PadLeft(8, '0');
        string b = Convert.ToString(p.B, 2).PadLeft(8, '0');
        string s1 = Convert.ToString((p.R ^ p.B), 2).PadLeft(8, '0');
        string s2 = Convert.ToString((p.G ^ p.B), 2).PadLeft(8, '0');
        char chk1 = s1[7];
        char chk2 = s2[7];

        if ((m1 == chk1) & (m2 == chk2))
        {
            ro = p.R; go = p.G; bo = p.B;
        }
        else if ((m1 != chk1) & (m2 == chk2))
        {
            go = p.G; bo = p.B;
            ro = (r[7] == '0') ?
                Convert.ToByte((r.Substring(0, 7) + "1"), 2) :
Convert.ToByte((r.Substring(0, 7) + "0"), 2);
        }
        else if ((m1 == chk1) & (m2 != chk2))
        {
            ro = p.R; bo = p.B;
            go = (g[7] == '0') ?
                Convert.ToByte((g.Substring(0, 7) + "1"), 2) :
Convert.ToByte((g.Substring(0, 7) + "0"), 2);
        }
        else
        {
            ro = p.R; go = p.G;
            bo = (b[7] == '0') ?
                Convert.ToByte((b.Substring(0, 7) + "1"), 2) :
Convert.ToByte((b.Substring(0, 7) + "0"), 2);
        }
    }
```

EK-3. K5 Steganografi uygulamasının veri ayıklama modülü için C# kaynak kodları

```
    private void buttonSelectStego_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofdStego = new OpenFileDialog();
        ofdStego.Filter = "Image Files (*.bmp, *.png, *.jpg) | *.bmp; *.png; *.jpg";
        ofdStego.InitialDirectory = @"C:\Users\Default\Desktop";
        if (ofdStego.ShowDialog() == DialogResult.OK)
        {
            textBoxStegoPath.Text = ofdStego.FileName.ToString();
            pictureBox1.ImageLocation = textBoxStegoPath.Text;
            label2.Text = "Status: Stego-Image is selected";
            stegoEtkin = true;
            if (stegoEtkin & pswExtractEtkin)
                buttonExtract.Enabled = true;
        }
    }
    private void buttonPswExtract_Click(object sender, EventArgs e)
    {
        while (true)
        {
            pswExtract = Microsoft.VisualBasic.Interaction.InputBox("Password must
be at least 2 characters!",
                "Enter Password for Extraction");
            if (pswExtract.Length >= 2) break;
        }
        label2.Text = "Status: Extraction password OK";
        pswExtractEtkin = true;
        if (stegoEtkin & pswExtractEtkin)
            buttonExtract.Enabled = true;
    }
    private void buttonExtract_Click(object sender, EventArgs e)
    {
        buttonExtract.Enabled = false;
        Bitmap img = new Bitmap(textBoxStegoPath.Text);
        int[] iarray, jarray;
        permStrad(img.Width, img.Height, pswExtract, out iarray, out jarray);
        int mesajboy = 0; string outtext = ""; bool oku = false;
        for (int i = 0; i < iarray.Length; i++)
            for (int j = 0; j < jarray.Length; j++)
                if (!oku)
                    for (int k = 0; k < 4; k++)
                    {
                        Color p = img.GetPixel(iarray[i], jarray[j] * 4 + k);
                        mesajboy += p.B;
                        if (p.B == 0) { oku = true; break; }
                    }
                else if (oku && outtext.Length < mesajboy)
                {
                    string hrfbts = "";
                    for (int k = 0; k < 4; k++)
                    {
                        Color np = img.GetPixel(iarray[i], jarray[j] * 4 + k);
                        int x1 = (np.R ^ np.B);
                        int x2 = (np.G ^ np.B);
                        string s1 = Convert.ToString(x1, 2).PadLeft(8, '0');
                        string s2 = Convert.ToString(x2, 2).PadLeft(8, '0');
                        string m1 = s1.Substring(7, 1).PadLeft(8, '0');
                        string m2 = s2.Substring(7, 1).PadLeft(8, '0');
                        hrfbts = hrfbts + m1[7] + m2[7];
                    }
```

EK-3 (devam). K5 Steganografi uygulamasının veri ayıklama modülü için C# kaynak kodları

```
                    byte value = Convert.ToByte(hrfbts, 2);
                    string harf = System.Text.Encoding.Default.GetString(new byte[]
{ value });
                    outtext = outtext + harf;
                }
                else
                {
                    j = jarray.Length - 1; i = iarray.Length - 1;
                }
        SaveFileDialog sfdOut = new SaveFileDialog();
        sfdOut.Filter = "Text Files (*.txt) | *.txt";
        sfdOut.InitialDirectory = @"C:\Users\Default\Desktop";
        if (sfdOut.ShowDialog() == DialogResult.OK)
        {
            textBoxOutPath.Text = sfdOut.FileName.ToString();
            using (StreamWriter file = new StreamWriter(textBoxOutPath.Text, false,
Encoding.Default))
            {
                file.Write(outtext);
            }
            label2.Text = "Status: Extraction completed... Ready";
            stegoEtkin = false; pswExtractEtkin = false;
        }
    }
```

EK-4. Ki-kare Steganaliz uygulamasının C# kaynak kodu

```
namespace Chi_Square_Steganalysis
{
    public partial class Form1 : Form
    {
        public Bitmap img;
        public int w, h, x = 0, y = 0;
        public int binsize;
        public Point[] points;
        public double[] redf = new double[256], redE = new double[256];
        public double[] grnf = new double[256], grnE = new double[256];
        public double[] bluf = new double[256], bluE = new double[256];
        public double[] pval; public char gom;

        public Form1()
        {
            InitializeComponent();
            textBoxImgPath.Enabled = false;
            buttonAtk.Enabled = false; buttonReset.Enabled = false;
        }
        public void nextpixel(out Color p)
        {
            p = img.GetPixel(x, y);
            if (x < w - 1) x++;
            else if (y < h - 1) { y++; x = 0; }
            else { y = 0; x = 0; }
        }
        private void buttonSelectImg_Click(object sender, EventArgs e)
        {
            labelWidth.Text = "Image Width: ";
            labelHeight.Text = "Image Height: ";
            labelPixel.Text = "# of Pixels: ";
            labelBin.Text = "Bin Size: ";

            OpenFileDialog ofdImg = new OpenFileDialog();
            ofdImg.Filter = "Image Files (*.bmp, *.png, *.jpg) | *.bmp; *.png;
*.jpg";
            ofdImg.InitialDirectory = @"C:\Users\Default\Desktop";

            if (ofdImg.ShowDialog() == DialogResult.OK)
            {
                textBoxImgPath.Text = ofdImg.FileName.ToString();
                img = new Bitmap(textBoxImgPath.Text);
                gom = alglib.getgom(textBoxImgPath.Text);

                w = img.Width; h = img.Height;
                binsize = (w * h) / 1000;

                points = new Point[(w * h / binsize) + ((w * h) % binsize)];
                pval = new double[(w * h / binsize) + ((w * h) % binsize)];

                labelWidth.Text += Convert.ToString(w);
                labelHeight.Text += Convert.ToString(h);
                labelPixel.Text += Convert.ToString(w * h);
                labelBin.Text += Convert.ToString(binsize);

                buttonAtk.Enabled = true;
            }
        }
```

EK-4 (devam). Ki-kare Steganaliz uygulamasının C# kaynak kodu

```
        private void buttonAtk_Click_1(object sender, EventArgs e)
        {
            buttonAtk.Enabled = false; buttonSelectImg.Enabled = false;
            x = 0; y = 0; int say = 0;
            for (int bin = 0; bin < w * h; bin += binsize)
            {
                Array.Clear(redf, 0, redf.Length); Array.Clear(redE, 0,
redE.Length);
                Array.Clear(grnf, 0, grnf.Length); Array.Clear(grnE, 0,
grnE.Length);
                Array.Clear(bluf, 0, bluf.Length); Array.Clear(bluE, 0,
bluE.Length);
                double dofred = 255, dofgrn = 255, dofblu = 255;
                for (int i = 0; i < binsize; i++)
                {
                    Color p = new Color();
                    nextpixel(out p);
                    redf[p.R]++; grnf[p.G]++; bluf[p.B]++;
                }
                for (int c = 0; c < 256; c++)
                {
                    redf[c] /= binsize; grnf[c] /= binsize; bluf[c] /= binsize;
                }
                double chi2red = 0, chi2grn = 0, chi2blu = 0;
                for (int j = 0; j < 128; j++)
                {
                    redE[2 * j] = (redf[2 * j] + redf[2 * j + 1]) / 2;
                    redE[2 * j + 1] = redE[2 * j];
                    if (redE[2 * j] != 0)
                    {
                        chi2red += (Math.Pow((redf[2 * j] - redE[2 * j]), 2)) /
redE[2 * j];
                        chi2red += (Math.Pow((redf[2 * j + 1] - redE[2 * j + 1]),
2)) / redE[2 * j + 1];
                    }
                    else dofred -= 2;

                    grnE[2 * j] = (grnf[2 * j] + grnf[2 * j + 1]) / 2;
                    grnE[2 * j + 1] = grnE[2 * j];
                    if (grnE[2 * j] != 0)
                    {
                        chi2grn += (Math.Pow((grnf[2 * j] - grnE[2 * j]), 2)) /
grnE[2 * j];
                        chi2grn += (Math.Pow((grnf[2 * j + 1] - grnE[2 * j + 1]),
2)) / grnE[2 * j + 1];
                    }
                    else dofgrn -= 2;

                    bluE[2 * j] = (bluf[2 * j] + bluf[2 * j + 1]) / 2;
                    bluE[2 * j + 1] = bluE[2 * j];
                    if (bluE[2 * j] != 0)
                    {
                        chi2blu += (Math.Pow((bluf[2 * j] - bluE[2 * j]), 2)) /
bluE[2 * j];
                        chi2blu += (Math.Pow((bluf[2 * j + 1] - bluE[2 * j + 1]),
2)) / bluE[2 * j + 1];
                    }
                    else dofblu -= 2;
                }
```

EK-4 (devam). Ki-kare Steganaliz uygulamasının C# kaynak kodu

```
                double redtest = alglib.chisquaredistribution(dofred, chi2red);
                double grntest = alglib.chisquaredistribution(dofgrn, chi2grn);
                double blutest = alglib.chisquaredistribution(dofblu, chi2blu);
                pval[say] = (redtest + grntest + blutest) / 3;
                double pv;
                Point dot = alglib.getdot(gom, w, h, say, bin, out pv);
                points[say] = dot; pval[say] = pv;
                if (say != 0) this.CreateGraphics().DrawLine(new Pen(Brushes.Black,
1), points[say - 1], points[say]);
                else this.CreateGraphics().DrawLine(new Pen(Brushes.Black, 1),
points[say], points[say]);
                say++;
            }
            string display = string.Join(" ", pval);
            MessageBox.Show("1 - p:\n" + display);
            buttonReset.Enabled = true;
        }

        private void buttonReset_Click(object sender, EventArgs e)
        {
            this.Invalidate();
            textBoxImgPath.Text = "";
            buttonSelectImg.Enabled = true;
        }
    }
}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	:	KOCATEPE, Ali İlker
Uyruğu	:	T.C.
Doğum tarihi ve yeri	:	12/08/1981 İzmit
Medeni hali	:	Bekâr
Telefon	:	0 (312) 202 38 44
e-posta	:	ailkerkocatepe@gazi.edu.tr

Eğitim Derecesi	Okul/Program	Mezuniyet yılı
Yüksek lisans	Gazi Üniversitesi Bilgisayar Bilimleri Anabilim Dalı	Devam Ediyor
Lisans	Gazi Üniversitesi Bilgisayar Öğretmenliği	2013

İş Deneyimi, Yıl	Çalıştığı Yer	Görev
2014-devam ediyor	Gazi Üniversitesi	Araştırma Görevlisi
2013-2014	Çorum/Bayat Ö.Mülazım Ortaokulu	Öğretmen
2007-2013	Trenkwalder/Vanlıoğlu A.Ş.	Ekip Amiri

Yabancı Dili

İngilizce

Yayınlar

1. Kocatepe, A. İ. ve Karacan, H. (2016). *Application of Different Steganography Algorithms on Various Image Formats*, International Conference on Applied Computing in Science & Engineering, Roma, İtalya.

2. Koşan, M. A., Kocatepe, A. İ. ve Karacan, H. (2016). *A Model of Steganographic Report for Web Application Security*, International Conference on Applied Computing in Science & Engineering, Roma, İtalya.

GAZİ GELECEKTİR...