KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL RESİM İÇERİSİNE VERİ GİZLEME UYGULAMALARI

YÜKSEK LİSANS TEZİ

Hilal GÜREL

Anabilim Dalı: Elektronik ve Bilgisayar Eğitimi

Danışman: Yrd. Doç. Dr. Celal ÇEKEN

KOCAELİ, 2006

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL RESİM İÇERİSİNE VERİ GİZLEME UYGULAMALARI

YÜKSEK LİSANS TEZİ

Hilâl GÜREL

Tezin Enstitüye Verildiği Tarih:

Tezin Savunulduğu Tarih:

Tez Danışmanı	Üye	Üye
Yrd.Doç.Dr. Celal ÇEKEN	Doç.Dr. İsmail ERTÜRK	Yrd.Doç.Dr. Hakan KAPTAN
(.....)	(.....)	(.....)

KOCAELİ, 2006

ÖNSÖZ VE TEŞEKKÜR

Güvenlik gibi güncel gereksinimler açısından veri gizleme üzerine yapılan çalışmalar, her geçen gün daha da büyük bir öneme sahip olmaktadır. Eskiden ilkel yöntemlerle yapılmaya çalışılan "Stenografi" (steganografik), günümüzde yazılım ile desteklenmekte ve güçlü algoritmalar kullanılmaktadır. Bu açıdan özellikle resim içerisine bilgi gömme teknikleri üzerine yoğunlaşılarak, bu alanda yeni yaklaşımlar elde edilmiştir.

Bu tezde sunulan araştırma ve çalışmalarda, veriyi gizlerken LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanarak resim üzerinde gözle görülür bir bozulma oluşmadan, UDP(User Datagram Protocol) /TCP (Transmission Control Protocol) protokollerinden yararlanarak kablosuz ortamda, gizli bilginin korunması ve güvenle istenen hedefe iletilmesi ve bant genişliğininetkin bir şekilde kullanılması hedeflenmiştir.

Tez çalışmalarım süresince değerli zamanlarını ayıran, bilgi ve deneyimlerini paylaşan, çalışmalarımı yönlendiren ve her zaman destek olan tez danışmanım sayın Yrd. Doç.Dr. Celal ÇEKEN'e (KO.Ü), tezin gerçekleştirilmesinde her türlü ilgi ve desteği gösteren Uğur ASAN'a, Yunus ÖZEN, Göksu ÖZEN çiftine ve Necla BANDIRMALI'ya teşekkürlerimi sunarım.

Beni bugünlerime getiren, her konuda destek veren ve yanımda olan çok değerli annem Fatma ve babam İsmail GÜREL 'e yaptıkları her şey için çok teşekkür ediyorum.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER

A	: Gizli mesajın ASCII kodunun bölümü
a	: p 'den üretilen asal bir kök
B	: Blue (Mavi)
G	: Green (Yeşil)
l	: İçine veri gizlenecek resmin büyüklüğü
m	: Gizlenecek metnin uzunluğu
M	: Bir resimdeki sütün sayısı
N	: Bir resimdeki satır sayısı
p	: Fonksiyondaki büyük bir asal sayı
R	: Red (Kırmızı)
Y	: Mesajın i. bitinin resmin içinde saklanacagı pozisyonu

Alt indisler

i	: Gizlenecek mesajın bit indeksi

Kısaltmalar

AES	: Advanced Encryption Standard
ASCII	: The American Standard Code for Information Interchange
BMP	: Bitmap
CMYK	: Baskıda dört temel işlem rengi (Cyan, Magenta, Yellow, Black)
CT	: Scitex Continuous Tone
DES	: Data Encryption Standard
EPS	: Encapsulated PostScript
GIF	: Graphics Interchange Format: Grafik Değiştirme Biçimi
ISO	:International Organization for Standardization), Uluslararası Standartlar Teşkilatı
JPEG	: Joint Photographics Experts Group: Birleşik Fotoğraf Uzmanları Grubu
LSB	: Least Significant Bit: En Önemsiz Bit
LZW	: Lempel–Ziv–Welch
MSB	: Most Significant Bit
OS/2	: IBM ve Microsoft tarafından 1985 yılında geliştirilmeye başlanan grafik arayüze sahip PC işletim sistemidir.
PC	: Personel Computer
PCX	: Zsoft firması tarafından PC Paintbrush yazılımı için geliştirilmiş resim formatı
PS	: Post Script CAD formatları
RGB	: Red Green Blue (Kırmızı Yeşil Mavi)
RLE	: Run–Lenght–Encoding
RSA	: Rivest– Shamir– Adleman
RSAAES	: Advanced Encryption Standard
TCP	: Transmission Control Protocol (İletim denetimi protokolü)
TIFF	: Tagged Image File Format

TL	: Türk Lirası
TV	: Televizyon
UDP	: User Datagram Protocol (Kullanıcı veri birimi protokolü)
XCF	: Experimental Computing Facility
XPM	:Pixmap

SAYISAL RESİM İÇERİSİNE VERİ GİZLEME UYGULAMALARI

Hilal GÜREL

Anahtar Kelimeler: Veri gizleme, stegonagrafi, damgalama, LSB yöntemi

Özet: Güvenlik gibi güncel gereksinimler açısından veri gizleme üzerine yapılan çalışmalar her geçen gün önem kazanmaktadır. Artık askeri uygulamaların yanı sıra, özel sektöre ait birçok alanda da güvenlik nedeniyle veri gizleme yöntemlerine başvurulmaktadır.

Bu tez çalışmasında; LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanılarak iletilmek istenen bilginin gizlenmesi araştırılmıştır. Ayrıca TCP/UDP protokollerinden yararlanarak kablosuz ortamda, gizlenmiş bilginin güvenli bir şekilde istenen hedefe iletilmesi amaçlanmıştır. Kablosuz ortamın en büyük problemlerinden ikisi güvenlik ve sınırlı bant genişliğidir. Gerçekleştirilen bu çalışmalarla resim üzerinde gözle görülür bir bozulma oluşmadan gizli bilginin kablosuz ortamdan gönderilmesi ve böylece bant genişliğinin etkin bir şekilde kullanılması sağlanmıştır.

Çalışmada, üç değişik veri gizleme yöntemi, iki farklı başarım ölçütüne göre karşılaştırılmıştır. Bunlardan ilki boyuttur. Gömü verisinin boyutu ile örtü verisinin boyutu arasındaki oran kullanılan veri gizleme yöntemlerine göre farklılıklar göstermektedir. Bu bağlamda, en avantajlı yöntemin RGB–ASCII olduğu görülmektedir. RGB – ASCII yönteminde 3 bayt veriye 1 bayt, İki Bit yönteminde 8 bayt veriye 2 bayt ve LSB yönteminde ise 8 bayt veriye 1 bayt veri saklanabilmektedir. İkinci başarım ölçütü ise işlem zamanıdır. Veri gizleme ve gizli veriyi çözme süreleri göz önüne alındığında en kötü sonucu LSB yönteminin, en iyi sonucu is yine RGB – ASCII yönteminin verdiği gözlemlenmiştir. Gerçekleştirme açısından bakıldığında ise LSB yönteminin diğerlerine nazaran kolay olduğu tespit edilmiştir.

Araştırma kapsamında kullanılan yöntemler ve gizlenmiş bilginin iletilmesini sağlayan algoritmalar Java yazılım dili kullanılarak gerçekleştirilmiştir.

DATA HIDING APLICATIONS USING DIGITAL PICTURES

Hilal GÜREL

Keywords: Data hiding, stenography, watermarking, LSB method

Abstract: Considering the security aspect, the importance of the studies on data hiding has been growing every day. Nowadays, data hiding techniques are employed not only for military purposes but also for other areas of private sector.

In this thesis, information hiding using LSB, Two Bits and an ASCII Code Hiding in One Pixel (RGB–ASCII) has been investigated. Besides, it is intended to securely transmit the hidden information to destination securely using TCP/UDP protocols. Two major drawbacks of the wireless medium are security and limited bandwidth. Using the algorithms implemented in these research studies, the data is embedded into an image called cover object, which results in a stego object. Ideally, the stego object is indistinguishable from the original image, appearing as if no additional information has been encoded. The stego object then is transmitted over wireless medium, thus the bandwidth efficiency is provided.

There are two performance metrics used for comparative analysis of information hiding techniques in this thesis. The first one is size metric. The ratio between embedded data size and cover data size differs according to the information hiding technique utilized. In this context, the best technique is the RGB–ASCII. The cover data and embedded data ratios of RGB–ASCII, Two Bits and LSB techniques are 3/1, 8/2 and 8/1 respectively. The second metric is the time elapsed. Considering the duration of both data hiding and extracting hidden data from the stego object, it is observed that the LSB technique has the worst result while RGB–ASCII has the best.

The algorithms utilized in this research study have been implemented using Java Programming Language.

1. GİRİŞ

Tam olarak “kaplanmış yazı” anlamına gelen Steganograpy’nin amacı önemli mesaj ya da bilginin varlığını saklamaktır. Bu yaklaşım bir nesnenin içerisine bir verinin gizlenmesi olarak tarif edilebilir. Ses, sayısal resim ya da video görüntüleri içerisine herhangi bir veri gömülebilir. Bu veriler düz metin dosyası olabileceği gibi herhangi bir görüntü içerisine başka bir görüntüyü de gizlemek mümkündür.

Günümüzde bilgisayar haberleşmesi ile anılan Steganografik, aslında geçmişi çok uzun yıllar öncesine dayanan bir bilgi gizleme yöntemidir. Örneğin; eski Yunan ve Roma’da M.Ö. 5. yüzyılda Susa kralı Darius tarafından göz hapsine alınan Histiaeus’un Milet’deki oğlu Aristagoras’a gizli bir mesaj göndermesi gerekmektedir. Histiaeus, kölelerinden birinin saçlarını kazıtır ve mesajını dövme şeklinde kölenin kafasına işletir. Kölenin saçları tekrar uzayınca onu Milet’e oğlunun yanına gönderir [1]. Bu, gizli yazma sanatı steganografik’nin ilk kullanıldığı yerlerden biridir. Bir başka örnek ise; eski Romalılar’ın ve Yunanlılar’ın gizli mesajları tabletlere yazmaları ve sonra tabletleri bal mumu ile kaplatmalarıdır. Böylece güvenli bir şekilde iletilmesi gereken bilgi gizlenmiş olurdu. Mesajı alanlar daha sonra bal mumunu kazıyarak iletilmek istenen bilgiye ulaşırlardı.

Şifreleme (kriptografi); bir iletinin içeriğini, uygun bilgi (anahtar bilgisi) elde olmadan okunamayacak hale getirme işlemidir. Şifrelemenin amacı, iletinin istenmeyen şahıslar tarafından okunmasını engellemektir. Şifre çözümü (deşifre) ise şifrelemenin tam tersi, yani şifreli metnin düz metine çevrilmesi işlemidir.

Şifrelemeyi, eskiden sadece askerler kullanırken günümüzde artık çok sayıda kişi ve kurum tarafından kullanılmakta ve hatta kullanımı zorunlu hissedilmektedir. Kriptografi Latince’den türemiş bir kelimedir. "Kryptos" gizli, "graphia" ise yazma anlamına gelmektedir. Bir başka deyişle kriptografi, gizli yazım sanatıdır. Kriptografi, bir mesajı şifreleyerek onun üçüncü şahıslar tarafından anlaşılabilirliğini

ve çözülebilirliğini engelleyerek iki taraf arasında güvenli bir iletişim sağlamayı amaçlar.

Steganografi de bir başka yazma yöntemidir. Latince de "steganos" görünmeyen anlamına gelmektedir. Steganografi aslında şifrelemenin alternatifi değil onun tamamlayıcısıdır.

Steganografi son yıllarda yeni bir şifreleme metodu olarak karşımıza çıkmaktadır. Bu yaklaşım kısaca bir nesnenin içerisine bir verinin gizlenmesi olarak tarif edilebilir. Bugünün steganografi yöntemleri güvenliği daha da arttırmak için şifrelenmiş verileri gizlemek için genelde ses, sayısal resim veya video dosyalarını kullanmaktadır. Şifrelenmiş veriler kendi başlarına insanların dikkatini çekebilir; fakat görüntü ya da ses dosyalarının içine gizlenmiş olduklarında hiç kimse fark etmeyeceğinden kırılmaya da çalışılmayacaktır.

Steganografik yöntemler aynı zamanda 1. ve 2. Dünya Savaşlarında da kullanılmıştır. Kimyagerler gizli bir mürekkep geliştirmişlerdir. Bu mürekkep ile zararsız görünen bir mektup, satırları arasına yazılmış birçok gizli mesaj içermektedir.

Belgeler de kendi içerisinde bilgiyi saklayabilirler. Mesaj açıkta gönderilir ve üçüncü şahıs bunun zararsız bir mesaj olduğu düşünür. Örneğin; aşağıdaki mesaj 2. Dünya Savaşında Alman bir casus tarafından gönderilmiştir.

“Apparently neutral’s protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by–product, ejecting suets and vegetable oils.”

Bu mesajdaki her kelimenin 2. harfini alırsak karşımıza “Pershing sails from NY June1” şeklinde bir gizli mesaj ortaya çıkacaktır.

Bilgi güvenliği günümüzde kişisel olarak önemli olsa da, bundan daha önemlisi bazı toplumların geleceklerinin teminatı olan özel iş ve görev yapan birimler/kurumlar için çok daha önem arz etmektedir. Askeri uygulamalar bunların başında

gelmektedir. II. Dünya Savaşı'ndan 35 yıl sonra açıklanan bir rapora göre, İngiltere ile Almanya arasında geçen savaşın seyri, savaş taktiklerini içeren mesajların çözülmesiyle değişmiş ve savaşın müttefikler tarafından kazanılmasında büyük rol oynamıştır [2]. Boston Globe gazetesi bir haberinde bu olayın, II. Dünya Savaşı hakkında daha önce yazılmış tarih kitaplarında birçok değişiklikler yapılmasını gerektirecek kadar önemli olduğunu vurgulamıştır. Tarihi şekillendirecek kadar önemli olan şifre çözme olayı, bu konunun önemini vurgulamada verilebilecek en iyi örneklerden birisidir. Günümüzde insanı gerçekten hayrete düşürecek milyonlarca örnek bulunabilir. Bilişim dünyasında sayısallaşmanın hızla yaygınlaşması, gerek kişisel gerekse kurumsal veri güvenliği için şifreleme metotlarının veya kriptolama sistemlerinin kullanımını bir zorunluluk haline getirmiştir.

Steganografi, kriptografi yöntemine yakın olmasına rağmen küçük de olsa farklılıklar içermektedir. Kriptografide amaç, mesajın içeriğinin saklanması iken steganografi de ise mesajın varlığını saklamaktır. Kriptografi işleminde veriyi gizlemek amacıyla bir kriptografik dönüşüm yapılır. Steganografide ise verinin saklandığı hiçbir şekilde anlaşılmamalıdır. İki yöntem önce gizli mesajın şifrelenmesi ve sonra da steganografik yöntemlerle saklanması şeklinde birleştirilebilir.

Damgalama, saklı mesajın varlığını gizlemeyi amaçlayan bir bilgi saklama yöntemidir. Steganografide ise mesaj gizlenerek varlığından istenmeyen kişilerin haberdar olmaması sağlanarak güvenli iletişim gerçekleştirilir. Modern steganografide, gizlenecek mesaj yani "gömü verisi" onu saklayacak bir "örtü verisi"nin (cover data) içine gömülür ve örtü verisinin insan algılamasıyla fark edilemeyecek ölçüde bozulmasına izin verilir. Damgalama uygulamalarında, gizlenecek mesaj "örtü verisi"nin bir gizlenmiş özelliği olarak bulunur ve "örtü verisi" aslında ticari bir müzik veya yazılım ürünü olduğu için bu "örtü verisi"nin bozulması istenmez.

Stegonagrafik yöntemlerin ticari uygulamaları dijital damgalama olarak bilinir. Bu yeni kavram steganografik ile karıştırılmamalıdır. Damgalama'nın amacı bir ses ya da görüntü dosyasının bazı özel modifikasyonlarla saklanması değil bir kişiye ait olduğunu belirtmektir. Yapılan değişimlerin steganografik deki gibi fark edilemez ve

güçlü olması gerekmektedir. Hiç kimsenin daha önceden işaretlenmiş bir dosyadaki işareti kaldıramaması ve kendi işaretini koyamaması hedeflenir.

Bugünün steganografi yöntemleri güvenliği daha da arttırmak amacıyla şifrelenmiş verileri gizlemek için genelde görsel ya da ses dosyalarını kullanmaktadır [3].

1.1. Damgalama

Bandrol, çek gibi değerli kağıtlar üzerinde uzun yıllardır kullanılan "watermark" teknolojisinin dijital ortamdaki karşılığı olan "digital damgalama", fotoğraf, video, resim vb gibi görsel içeriğe saklanan ve insan gözü ile seçilemeyen özel bir mesajdır [4].

Damgalama, Steganografi'nin kullanım araçlarından biridir. Dijital ortamda resim, ses, video ve doküman gibi tüm dosyalar kolayca kopyalanabilmekte, kullanılabilmekte ve dağıtılabilmektedir. Bu kopyalama işlemi dijital ortamın yapısı gereği orijinal dosyanın aynı kalitede bir kopyasını oluşturmaktadır. Bu yüzden bir yazar ya da fotoğrafçı yazdığı yazının ya da çektiği resmin böyle bir dijital ortamda telif hakkı için bir girişimde bulunabilmesi çok zordur. Bunun için dosyaya ekstra bir bilgi eklemek ve sadece bu bilgiyi içeren dosyayı dağıtmak yeterli bir çözümdür. Damga olarak bilinen bu gömülen bilgi; dosya hakkında sahiplik, telif hakkı veya lisans bilgisi sağlar. Sonuç olarak bir dosyanın (metin, ses, resim, video) telif hakkını sağlayabilmek için o dosyaya bir bilgi gömme işlemine damgalama, bu bilgiye de damga denmektedir.

Dijital damgalar ikiye ayrılır:

1. Görünür damga (visible watermark)
2. Görünmez damga (invisible watermark)

Görünen damgalar insan gözünün rahatlıkla algılayabileceği izlerdir. Paralarda bulunan, ışığa tutunca görünen resimler (Türk Lira'sındaki Atatürk resmi gibi), ya da

televizyon kanallarında o görüntünün hangi kanal ya da ajans tarafından çekildiğini gösteren ekranın köşesinde bulunan bir logo, görünen damgalara örnek verilebilir.

Görünmeyen damgaya bir örnek ise; pasaportlarda bulunan kişiye ait seri numarası fotoğrafın içerisine gömülmesidir. Herhangi biri elde ettiği bir pasaporta kendi resmini yapıştırdığı zaman özel tarayıcılarla fotoğraf tarandığında seri numarasının tutmadığı ya da olmadığı saptanabilir.

Görünmeyen damgaların görünenlere göre bazı avantajları vardır. Görünmeyen damgalarda, damga yerleri belli değildir ya da damga olup olmadığı fark edilmeyebilir. Damgayı tüm resim içine dağıtmak genel bir uygulamadır. Bu, resmi kesme saldırılarına (cropping attacks) karşı biraz olsun koruma sağlar. Fakat dosya içerisine gömülecek olan bilgi ne kadar az ise saldırılara karşı o kadar güçlü ve güvenli olur.

Steganografinin diğer bir şekli, damga oluşturma olarak adlandırılmaktadır ve ticari alanda gittikçe artan bir hızla kullanılmaktadır [5]. Damga oluşturulurken taşıyıcı ortam çok fazla bir değişikliğe uğratılmadan küçük bir bilgi gömülür. Damga oluşturma genellikle, telif hakkı alınan web sayfaları ya da ses dosyaları gibi dijital ortamların korunmasında kullanılır.

Damga oluşturma ve steganografi arasındaki tek fark şudur: damga oluşturmada kılıf, haberleşme nesnesiyken steganografide, gizlenen mesaj haberleşme nesnesidir [6]. Dijital damgaların bir kısmı insan gözüyle görülebilirken diğer kısmı görülmeyebilir. Görülebilen damgalar, genellikle görüntünün küçük bir alanıyla sınırlandırılmıştır. Bu damgaları kaldırmak isteyen kişiler, resimlerin ilgili kısmını keserek rahatlıkla kaldırabilir. Bu sebeple görünebilen damgalar yöntem olarak steganografi şeklinde değerlendirilmezler. Bu durumda görünmeyen damgalar, görünen damgalarlara göre daha fazla avantaj sağlamaktadır; çünkü nerede bulundukları bilinmemektedir.

Literatürde farklı steganografi yöntemleri üzerine birçok çalışma/araştırma bulunmaktadır. Aşağıdaki Bölüm 1.2'de bu çalışmalardan birkaçı kısaca özetlenmektedir. Bu çalışmalara Bölüm 4'de detaylı bir şekilde değinilmektedir.

1.2. Veri Gizleme Yöntemleri

Veri gizleme yöntemlerinin ilk uygulamalarından biri LSB Yöntemidir. Resim dosyalarına bakıldığında; herhangi bir pikseli oluşturan bitlerden, en az önemli olan, en az anlam taşıyan bitler (Least Significant Bit – LSB) üzerinde yapılacak bir modifikasyon, algılama farkıyla sonuçlanacak kadar etkili bir renk değişikliğine neden olmaz.

Bu yöntemde; resmi oluşturan her pikselin her bayt'nın en önemsiz biti (son biti)değiştirilerek yerine gizlenmesini istediğimiz verinin bitleri sırasıyla verinin başlangıcından itibaren birer birer yerleştirilmektedir. Burada her sekiz bitin en fazla bir biti değişikliğe uğratıldığından ve eğer değişiklik olmuşsa da değişiklik yapılan bit'in o bayt'ın en az anlamlı bit'i olmasından dolayı, ortaya çıkan steganogramdaki (örtü verisi + gömü verisi) değişimler insan tarafından algılanamaz boyutta olmaktadır.

Örneğin 11000000 ile 11000001 ikili sistemlerini karşılayan renkler, insan gözüne aynı renk olarak görünür ve bunun ayırt edilebilmesi 0'a yakın bir ihtimaldir. İletilmek istenilen gizli mesaj bu bitlere gömülebilir.

(a)
Şekil 1.1:a) Orijinal resim

(b)
b)Veri gömülü resim

LSB uygulamasına Şekil 1.1'de resimler bir örnektir. Şekil 1.1a'daki resim hiçbir işlem yapılmamış orijinal resimdir. Daha sonra orijinal resme LSB yöntemi ile veri gizlenmiştir. Şekil 1.1b'deki resim, içersine veri saklanmış bir resimdir. Resimler

karşılaştırıldığında, aralarındaki farkın gözle görülemeyecek boyutta olduğu ortadadır.

İçerisine veri gizlenmiş resim sıradan bir mesajla birlikte e–posta veya başka bir elektronik iletişim yoluyla gönderildiğinde, internet trafiğini denetleyen veya mesajı alan herhangi bir kişi sadece sıradan mesajı okuyabilecek ve mesaj ilişiğindeki resmi görebilecektir.

Herhangi bir resim üzerine gizlenen doküman, resim boyutunu değiştirmemektedir. Boyut değişikliğinin olmaması şifreleme işleminin başarısının ayrı bir göstergesidir. Gizlenen verinin tekrar elde edilmesi için yine geliştirilen yazılım ile işlemi tersine gerçekleştirmek ve LSB bitlerini okuyarak dokümanı geri elde etmek mümkündür.

Bu yaklaşımın tek bir dezavantajı gönderilecek mesajın uzunluğunun resim boyutuna bağlı olmasıdır. Uzun mesajlar için yüksek çözünürlükte ve boyutta resimlerin kullanılması gerekir.

Veri gizleme yöntemlerinden bir diğeri Bir piksel içerisine bir ASCII kodunun gömülmesi yöntemidir. Burada amaçlanan resim üzerinde çok büyük bozulmaya yol açmadan en büyük miktarda bilginin resim içerisine gömülmesidir. Bu uygulamada her bir piksele bir ASCII kodu gömülebilmekte ve de sonuçta 1 Bayt bilgi saklanmaktadır.

Bu yöntemde LSB yöntemine göre daha fazla bilgi gömülebilmektedir. Bu yönüyle daha avantajlıdır. LSB yöntemindeki gibi resimde oluşan bozulmaları gözün algılayamaması bu durumu önemsiz kılmaktadır.

Veri saklamada önemli olan orijinal piksel ağırlıklarının olabildiğince düşük oranlarda değişikliğe uğramasıdır. Bir resimde R'nin ağırlığı G ve B'nin ağırlığından farklıdır. Buradan yola çıkılarak "R" kodlama ağırlığının değiştirilmesi yöntemi geliştirilmiştir.

“R” ağırlığına gömülen ASCII kodunun ilk biti (MSB) olup değer olarak 0,1 ve 2 değerlerini almaktadır [7]. ASCII kod karşılıkları 0–255 arasında değerler aldığından ilk sayı değerinin 0, 1 ya da 2 olması gerekir. Bu durum bir avantaj olarak kullanılabilir: RGB ağırlıklı kodlama ile veri gizleme uygulamasında 8–bit olarak 3 ayrı renk RGB ağırlıklarını oluşturmaktadır.

Diğer bir yöntem olan Kısmi Optimizasyon Yöntemi ile veri gömme yöntemi ile resim 8–farklı bölgeye ayrılarak ilk önce 1. bölgeyi oluşturan piksellerin RGB ağırlıkları belirlenir. Eş zamanlı olarak gömülecek dosyanın ASCII kodları da belirlenmektedir. Bu bölgeye 8–farklı optimizasyon uygulanarak sonuçlar birbirleri ile karşılaştırılır. Orijinal piksellerle karşılaştırıldığında en az hata/bit oranına sahip olan durum tespit edilerek bir değişkende saklanır. Her bölge için bu işlemler tekrarlandığında, sonuçta tüm bölgelerin hangi optimizasyonla gömme işlemine tabi tutulduğunda daha az bozulma oluşacağı tespit edilir. Ardından tüm resim bölgeleri kaydedilen bu değişkenlere göre 8–bit ASCII kodlarını gömmek üzere düzenlenir.

Maskeleme ve filtreleme uygulaması ise görünür damgalama uygulamalarında kullanılan bir yöntemtir. Uygulama alanı gri tonlamalı resimler ve 24 bit BMP formatı ile sınırlıdır. Resimde görülebilir işaretleme yapmak için kullanılır.

LSB yöntemine alternatif olarak, Logaritma Kullanılarak Rasgele LSB Ekleme Yöntemi geliştirilmiştir.Sıralı LSB yönteminde verinin resmin satırlarına ya da sütunlarına sıra ile yerleştirilmesinden dolayı gizli mesajın elde edilmesi işlemi oldukça kolaydır. Bu yüzden verileri rasgele bir şekilde resmin içine saklamak daha güvenlidir.

1.3. Literatürde Yapılan Çalışmalar

Literatürde veri gizleme alanında, çok eski zamanlardan bu yana bir çok çalışma yapılmıştır. Aşağıda bunlardan birkaçı özetlenmiştir.

Trithemius (1462–1516) [8] çalışması steganografinin karmaşık bir sistemini içermektedir. Ayrıca bilgi bilimlerinin bir sentezini ve hafıza, sihir, hızlandırılmış dil

öğrenme sistemi ve mesajları sembolsüz veya ulak olmadan gönderme sistemleri ile ilgili bilgiler içermektedir.

Schotti G. (1665), [9] kitabında 400 sayfa Steganograpi ve kriptografiden bahsetmiştir.

J.D. hayhurst O.B.E. (1870–1871), [10] çalışmasında güvercinlerin ve mikro fotoğrafı'nın 1870–1871 Fransa–Prusya savaşı sırasında Paris'in işgali esnasında yaygın bir şekilde nasıl kullanıldığını anlatmaktadır.

Kerckhoffs A. (1883), [11] çalışmasında çoğunlukla kriptografi üzerinde durulmuş ve steganografik sistemlerin dizaynı ile ilgili bilgiler verilmiştir.

Briquet C. (1907) [12] damgalamanın tarihsel sözlüğüdür.

15. ve 16. Yüzyılda Şifreleme (1996) [13] çalışması 15. ve 16. yüzyıllarda kullanılan steganografik ve şifreleme sistemleri anlatmaktadır.

Kelly T. (1998) [14] kitabında Yunanca bir kelime olan skytale'in bir enkripsiyon cihazı olmadığını savunmaktadır, ancak asıl metnin mesaj olarak yazılmış bir parça materyal taşımayı kolaylaştırmak amaçlı bir tahta parçası kullanıldığını ortaya çıkarmaktadır.

Reeds J.(1998) [15] kitabında birçok sayı tablosunun, sihirli hecelemelerle tersine çevrilmiş bir alfabe kullanılarak Mod 25'e göre çözümlenmesiyle deşifre edilebileceği anlatılıyor.

1.4. Tez Çalışmasının Amacı ve Önerilen Çözüm Yöntemi

Bu tez çalışmasının amacı kısaca şöyle özetlenebilir;

- LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanarak BMP formatındaki bir resim içerisine herhangi bir metin dosyasındaki veriyi gizleyerek yeni bir resim dosyası oluşturmak.

- Gizlenmiş bilgiyi içeren resim dosyasını kablosuz ortamdan TCP ve UDP protokollerini kullanarak göndermek.
- Gizlenmiş bilgiyi içeren resim dosyasından iletilmesi istenilen bilgiyi almak.
- Gönderilmek üzere oluşturulan ve gizli bilgiyi de içeren yeni resim dosyasında gözün algılayabileceği herhangi bir değişikliğin olmamasını sağlamak.
- LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemlerinin başarımlarını karşılaştırmalı olarak incelemek.

1.5. Tez Çalışmasının Katkıları

Yapılan tez çalışmasının bilime ve endüstriye kazandırdığı yenilikler ve katkı aşağıda maddeler halinde sunulmaktadır:

- LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanarak BMP formatındaki bir resim içerisine herhangi bir metin dosyasındaki veriler gizlenmiştir.
- Gizlenmiş bilgiyi içeren resim dosyası kablosuz ortamdan TCP ve UDP protokollerini kullanarak transfer edilmiştir.
- Gizlenmiş bilgiyi içeren resim dosyasından iletilmesini istenilen bilgi çıkarılmıştır.
- LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri zaman ve kapasite ölçütlerine göre karşılaştırılarak başarım değerlendirmesi yapılmıştır.

1.6. Tez Organizasyonu

Yapılan tez çalışması aşağıda kısaca açıklanan bölümlerden oluşmaktadır:

Bölüm 1: Giriş: Bu bölümde tez çalışmasına konu olan problemin tanımı, çalışmanın amacı, literatürde bu problemin çözümü üzerine yapılan çalışmaların özeti, tez çalışmasını literatürde yapılan çalışmalardan ayıran temel özellikler ve tez çalışmasında izlenilen yöntem ile tez organizasyonu hakkında bilgi sunulmaktadır.

Bölüm 2: Veri Gizleme / Gömme Yöntemleri: Bu bölümde şimdiye kadar veri gizleme alanında yapılmış çalışmalardan ayrıntılı olarak bahsedilmiştir.

Bölüm 3: Resim Formatları: Bu bölümde çeşitli görüntü formatlarından bahsedilmekte ve tez çalışmasının bir parçası olan BMP formatına ayrıntılı olarak değinilmektedir.

Bölüm 4: Resim İçerisine Veri Gizleme Uygulaması: Bu bölümde LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri, TCP ve UDP protokolleri hakkında bilgi verilmektedir. Bununla birlikte resim içerisine veri gizleme uygulamasının tasarım aşamalarından, LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri ile resim içerisine veri gizleme ve içerisine veri gizlenmiş dosyadan gömü verisinin çıkarılması uygulamasının akış diyagramları ayrıntılı olarak verilmektedir. LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanarak TCP/UDP protokolleri ile kablosuz ortamda gizli verinin iletilmesi ve alınması uygulaması programının ekran çıktıları ayrıntılı olarak sunulmuştur.

LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanarak TCP/UDP protokolleri ile kablosuz ortamda gizli verinin iletilmesi ve alınması uygulamasına ait program kodları Ek– A ve Ek –B'de verilmektedir.

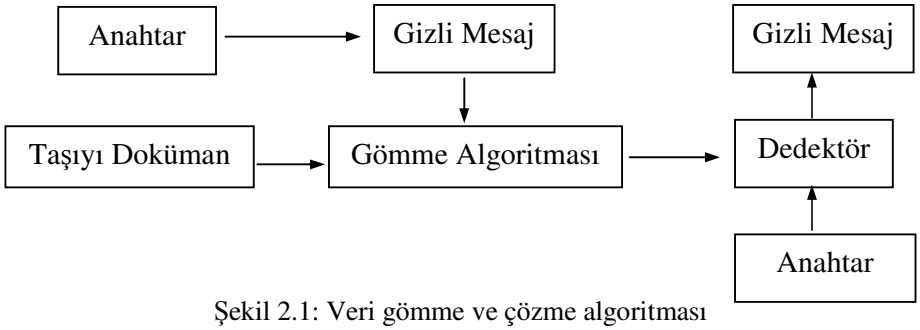
Sonuçlar ve Öneriler bölümünde, yapılan çalışmalardan elde edilen sonuçlar genel hatlarıyla değerlendirilerek çalışmanın bilim dünyasına sağlayabileceği katkılar tartışılmıştır. Daha sonra yapılabilecek çalışmalar için önerilerde bulunulmaktadır.

2. VERİ GİZLEME / GÖMME YÖNTEMLERİ

2.1. Giriş

Steganografi en önemli bilgi gizleme yöntemlerinden biridir. Bu yaklaşım ses, sayısal resim video görüntüleri gibi nesnelerin içerisine bir verinin gizlenmesi olarak tanımlanabilir.

Bütün veri gizleme yöntemleri Şekil 2.1'de görüldüğü gibi gömme algoritması ve bir detektör fonksiyonundan meydana gelmektedir. Gömme algoritması gizli mesajı bir taşıyıcı doküman içerisine gömmek için kullanılır. Gömme süreci bir anahtar kelime tarafından korunmaktadır ve bu yüzden yalnızca yetkili kişiler gizli anahtar kelime ile gizli mesaja erişebilirler.



Şekil 2.1: Veri gömme ve çözme algoritması

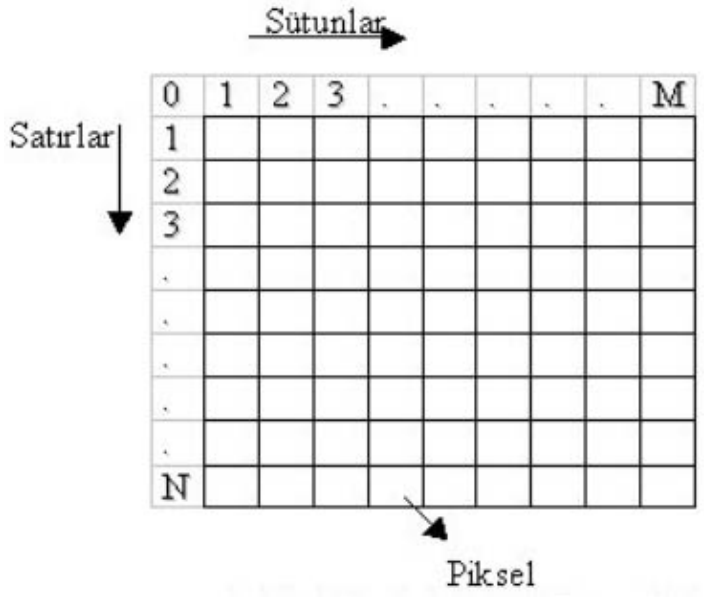
Bu yaklaşımda içine bilgi gizlenen artama örtü verisi (cover – data), oluşan ortama da stego – metin (stego – text) veya stego – nesnesi (stego –object) denmektedir [16].

2.2. Sayısal Resim

Sayısal resimler kare ebatlarında birçok resim noktalarından oluşurlar. Belirli renk bilgilerini barındıran bu noktalar, piksel olarak adlandırılır. Bunların adedi ne kadar fazla olursa fotoğraf o kadar ayrıntı zenginliğine sahip olur. Dijital fotoğraf makineleri gibi kayıt cihazlarında çözünürlük, ne kadar piksel algılandığını, ekranlarda ise, gösterilebilen nokta sayısı değeridir. Veriler, mutlak değerler olarak

gösterilir. Örneğin: Dijital fotoğraf makinelerinde 1600x1200 piksel, ekranlarda 1024x768 piksel gibi.

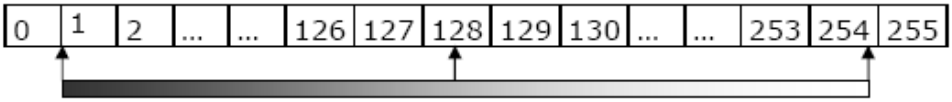
Steganografi kısaca bir nesnenin içerisine bir verinin gizlenmesi olarak tarif edilebilir. Ses, sayısal resim, video görüntüleri üzerine veri saklanabilir. Bu veriler metin dosyası olabileceği gibi, herhangi bir görüntü içerisine saklanabilecek başka bir görüntü dosyası da olabilir.



Şekil 2.2: Sayısal resim temel yapısı

Sayısal resimler Şekil 2.2'de gösterildiği gibi N satır ve M sütunluk bir dizi ile temsil edilir. Genellikle satır ve sütun indeksleri y ve x veya r ve c olarak gösterilebilir. Resim dizilerinin çoğu kare şeklindedir. Yani N=M ve tipik N ve M; 128, 256, 512 veya 1024 gibi değerler alabilmektedir. Bir resim dizisinin elemanlarına piksel denir. En basit durumda pikseller 0 veya 1 değerini alırlar. Bu piksellerden oluşan resimlere ikili (binary) resim denir. 1 ve 0 değerleri sırasıyla aydınlık ve karanlık bölgeleri veya nesne ve zemini (nesnenin önünde veya üzerinde bulunduğu çevre zemini) temsil ederler [17]. Resimlerin ışık seviyelerini daha iyi derecelendirebilmek için piksel başına 1 bayt kullanılır. Bununla 0 (siyah) ile 255 (beyaz) arasında tam sayılar elde edilebilir. Şekil 2.3'de de görüldüğü gibi bu sayılar arasındaki değerler gridir ve

bundan dolayı bir resme ait tam sayı "gri ton seviye" (gray level) olarak isimlendirilir.



Şekil 2.3: 1 x 256 boyutlarında gri tonlamada bir resim

Renkli resimlerde ise; renkler, kırmızı, yeşil ve mavi gibi 3'lü gruptan oluşan kümelerle ifade edilmektedir. Renklerin her biri için N x M lik bir diziye ihtiyaç duyulur. Sayısal değer büyüdükçe renk daha koyulaşmaktadır [18]. Üç temel renkten faydalanılarak istenilen renk oluşturulmaktadır. RGB (Kırmızı–Yeşil–Mavi) renk modelinde 256 x 256 x 256 = 16.777.216 adet değişik tonda renk ifade edilebilmektedir. Böylece belirli bir pozisyondaki pikselin Tablo 2.1'de görüldüğü gibi kırmızı, yeşil ve mavi değerleri resmin bileşenlerinin şiddetini belirler.

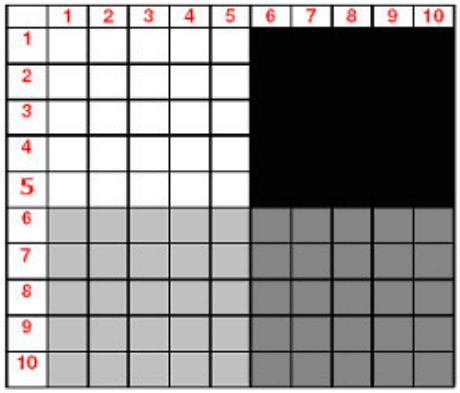
Tablo 2.1: Temel renklerin değerleri

TEMEL RENKLER	KIRMIZI	YESİL	MAVİ
Red (Kırmızı)	255	0	0
Green (Yesil)	0	255	0
Blue (Mavi)	0	0	255

2.3. LSB Yöntemi

Bilgisayar steganografisi iki temel prensip üzerine kurulmuştur. Bunlardan ilki sayısal hale getirilmiş resim veya ses dosyalarının, sahip oldukları fonksiyonlarını kaybetmeden değiştirilebilmeleri ilkesidir. İkincisi ise, insanın, renk veya ses kalitesinde meydana gelen ufak değişiklikleri ayırt edememesidir. Bunun amacı gereksiz bilgiler taşıyan nesnelerin içindeki bilgileri, başka bilgi parçacıklarıyla yer değiştirmektir. Resim dosyalarına bakacak olursak; herhangi bir pikseli oluşturan bitlerden en az anlam taşıyan bitler üzerinde yapılacak bir değişiklik, algılama farkıyla sonuçlanacak kadar etkili bir renk değişikliğine neden olmaz.

Sadece gri seviyeli renklerden oluşmuş, her pikselin 8 bite karşılık geldiği bir resim Şekil 2.4'de görülmektedir.



	1	2	3	4	5	6	7	8	9	10
1	255	255	255	255	255	0	0	0	0	0
2	255	255	255	255	255	0	0	0	0	0
3	255	255	255	255	255	0	0	0	0	0
4	255	255	255	255	255	0	0	0	0	0
5	255	255	255	255	255	0	0	0	0	0
6	192	192	192	192	192	128	128	128	128	128
7	192	192	192	192	192	128	128	128	128	128
8	192	192	192	192	192	128	128	128	128	128
9	192	192	192	192	192	128	128	128	128	128
10	192	192	192	192	192	128	128	128	128	128

(a) (b)

Şekil 2.4: a) 10 x 10 piksel boyutlarında gri tonlardan (0–255) oluşmuş bir resim b) Gri renk tonlarındaki bu resmin piksellerinin sayısal değerleri

Şekil 2.4a'nın 1. satırına bakacak olursak Şekil 2.5'de görüldüğü gibi resmin 1–5.inci piksellerin gri tonlarının değerleri 255, 6–10. piksellerinki ise 0'dır.

	1	2	3	4	5	6	7	8	9	10
1	255	255	255	255	255	0	0	0	0	0

Şekil 2.5: Orjinal resmin 1. satırı

Aşağıda bu değerlerin 8–bitlik karşılıkları gösterilmiştir.

$$255=(1x2^7)+(1x2^6)+(1x2^5)+(1x2^4)+(1x2^3)+(1x2^2)+(1x2^1)+(1x2^0)=11111111 \quad (2.1)$$

$$0=(0x2^7)+(0x2^6)+(0x2^5)+(0x2^4)+(0x2^3)+(0x2^2)+(0x2^1)+(0x2^0)=00000000 \quad (2.2)$$

Şekil 2.4b'nin 7. satırına bakacak olursak Şekil 2.6'de görüldüğü gibi resmin 1–5. piksellerin gri tonlarının değerleri 192, 6–10. piksellerinki ise 128'dir.

	1	2	3	4	5	6	7	8	9	10
7	192	192	192	192	192	128	128	128	128	128

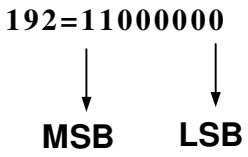
Şekil 2.6: Orjinal resmin 7. satırı

Aşağıda bu değerlerin 8–bitlik karşılıkları gösterilmiştir.

$$192=(1x2^7)+(1x2^6)+(0x2^5)+(0x2^4)+(0x2^3)+(0x2^2)+(0x2^1)+(0x2^0)=11000000 \quad (2.3)$$

$$128=(1x2^7)+(0x2^6)+(0x2^5)+(0x2^4)+(0x2^3)+(0x2^2)+(0x2^1)+(0x2^0)=10000000 \quad (2.4)$$

8–bit lik yapıda en önemli, anlamlı bitler (Most Significant Bits – MSB) solda ve en az önemli, anlamlı bitler sağda yer almaktadır. Şekil 2.7'de 192 sayısının LSB ve MSB bitleri gösterilmiştir.



Şekil 2.7: 192 sayısının MSB ve LSB bitlerinin gösterimi

Eğer MSB değiştirilirse bunun renk üzerinde büyük bir etkisi olacaktır. LSB değiştirilirse bunun etkisi gözün algılayamayacağı kadar çok küçük olacaktır.

192 sayısını ele alındığında 192 sayısının MSB biti 1'dir. Bu değer 0 yapıldığında sayının değeri 64 olacaktır. Bu çok büyük bir farktır. Tam tersini MSB yerine LSB değiştirilirse yani 0 olan LSB biti 1 yapılırsa sayının yeni değeri 193 olacaktır. Bu ise çok küçük bir farktır. Bir resim için bu durum düşünüldüğünde gözün bunu algılaması imkansızdır. MSB biti değiştirildiğinde ise renkte çok büyük fark olacağından göz bunu çok rahat algılayacaktır. Renkte ve resimde bozulmalar olacaktır.

Bitler soldan sağa doğru, genelden özele gider ve tanımlayıcı kimlik kazanır. Bu yöntem resim üzerine uygulanırsa ve resim üzerinde LSB değiştirilirse, bunun çok küçük bir etkisi olacağı bellidir. İnsan gözü bahsedilen ilk 7 bit civarında bir algı kapasitesine sahiptir, yani ilk 7 bitin yerini tutan renkleri seçebilir ve ayırt edebilir. Diğer bir deyişle, son biti kapsayan bir renk değişimi gerçekleştiğinde, bu olay normal biri için pek bir şey ifade etmez; çünkü insan gözü bu alana denk düşen renkleri algılayamaz.

Örneğin; 11000000 ile 11000001 ikili sistemlerini karşılayan renkler, insan gözüne aynı renk olarak görünür ve bunun ayırt edilebilmesi 0'a yakın bir ihtimaldir. Birilerine iletilmek istenilen gizli bir mesaj bu bitlere çok rahatlıkla gömülebilinir.

Şekil 2. 4a'da ki resim içerisine "HSC75" gibi metinden oluşmuş bir şifre bilgisini gizleyelim. Bilindiği gibi klavyede bulunan her karakterin bir ASCII kod karşılığı vardır. Tablo 2.2'de örnek metnimizde bulunan H, S, C, 7, 5 karakterlerinin ASCII kod ve ikilik sistemdeki karşılıkları verilmiştir.

Tablo 2.2: H, S, C, 7, 5 karakterlerinin ASCII kod ve ikilik sistemdeki karşılıkları

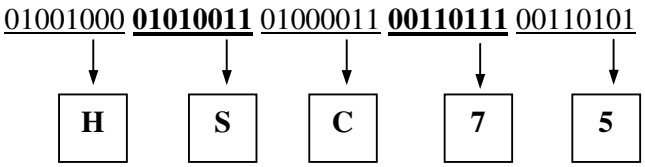
KARAKTER	**ASCII KODU**	**KODUN İKİLİK KARŞILIĞI**
H	72	01001000
S	83	01010011
C	67	01000011
7	55	00110111
5	53	00110101

Orjinal resmin piksellerinin 8–bit'lik ikili yapıda gösterimi Tablo 2.3'de verilmiştir.

Tablo 2.3: Orjinal resmin piksellerinin 8–bit'lik ikili yapıda gösterimi

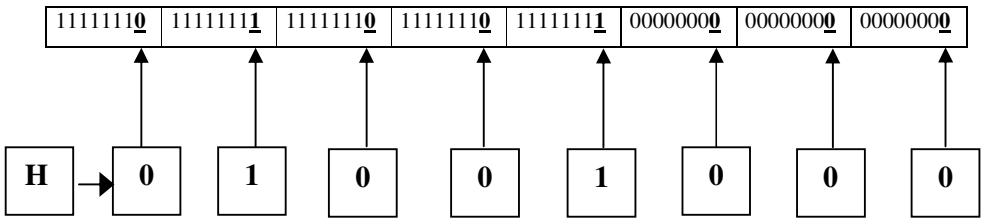
11111111	11111111	11111111	11111111	11111111	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	00000000	00000000	00000000	00000000	00000000
11111111	11111111	11111111	11111111	11111111	00000000	00000000	00000000	00000000	00000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000

Şifremizin de Tablo 2.2'deki ikilik sistemdeki karşılıklarını bir araya getirilirse; Şekil 2.8'de gösterildiği gibi 0 ve 1'lerden oluşmuş bir ikili sayı dizisi elde edilir.



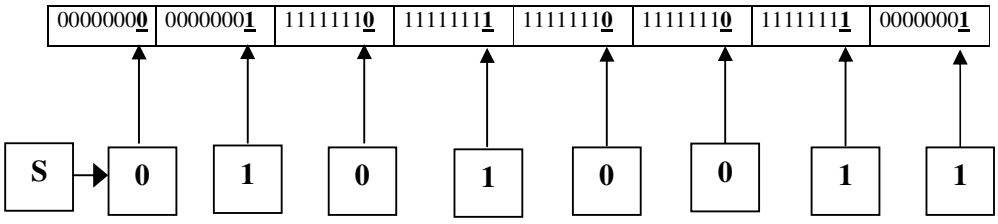
Şekil 2.8: H, S,C.7.8 karakterlerinin ikili karşılıkları

"H" karakterinin 01001000 bitlerini resmin ilk 8 pikselinin ikilik değerlerinin LSB'lerine yerleştirilmiş durumu Şekil 2.9'da gösterilmiştir.



Şekil 2.9: "H" karakterinin bit değerlerinin, resmin ilk 8 pikselinin ikilik değerlerinin LSB'lerine yerleştirilmesi

Şekil 2.10'da "S" karakterinin 01010011 bitlerini resmin H'yi yerleştirdiğimiz piksellerin devamında gelen diğer 8 pikselinin ikilik değerlerinin LSB'lerine yerleştirilmiş hali gösterilmektedir.



Şekil 2.10: "S" karakterinin bit değerlerinin, resmin 9–16. pikselinin ikilik değerlerinin LSB'lerine yerleştirilmesi

Bu şekilde yerleştirme işlemini, resmin içerisine saklanacak metinin tüm karakterlerinin bit değerleri için sıra ile devam edildiğinde sonuçta oluşacak metin gizlenmiş resmin piksel değerleri Tablo 2.4'deki gibi olacaktır.

Tablo 2.4: Orijinal resmin içerisine metin gizlenmiş resmin piksel değerleri

11111110	11111111	11111110	11111110	11111111	00000000	00000000	00000000	00000000	00000001
11111110	11111111	11111110	11111110	11111111	00000001	00000000	00000001	00000000	00000000
11111110	11111110	11111111	11111111	11111110	00000000	00000001	00000001	00000000	00000001
11111111	11111111	11111110	11111110	11111111	00000001	00000000	00000001	00000000	00000001
1111111	1111111	1111111	1111111	1111111	0000000	0000000	0000000	00000000	00000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000
11000000	11000000	11000000	11000000	11000000	10000000	10000000	10000000	10000000	10000000

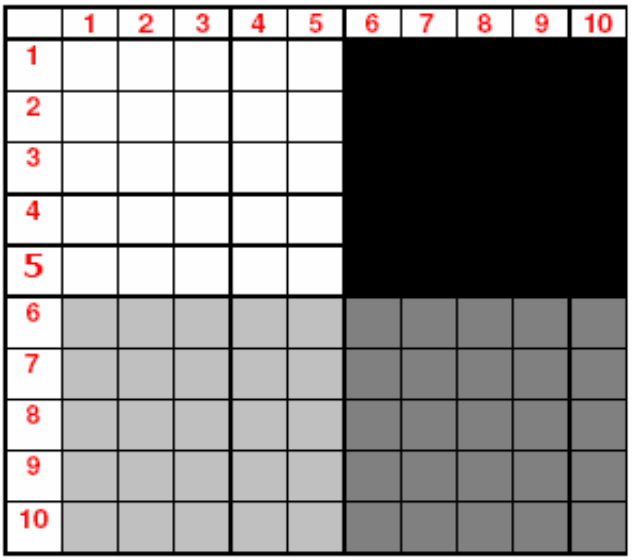
Tablo 2.5: Orijinal resmin piksel değerleri ASCII karşılıkları

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	255	255	255	255	255	0	0	0	0	0
2	255	255	255	255	255	0	0	0	0	0
3	255	255	255	255	255	0	0	0	0	0
4	255	255	255	255	255	0	0	0	0	0
5	255	255	255	255	255	0	0	0	0	0
6	192	192	192	192	192	128	128	128	128	128
7	192	192	192	192	192	128	128	128	128	128
8	192	192	192	192	192	128	128	128	128	128
9	192	192	192	192	192	128	128	128	128	128
10	192	192	192	192	192	128	128	128	128	128

Tablo 2.6: İçerisine metin gizlenmiş resmin piksel değerlerinin ASCII karşılıkları

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	254	255	254	254	255	0	0	0	0	1
2	254	255	254	254	255	1	0	1	0	0
3	254	254	255	255	254	0	1	1	0	1
4	255	255	254	254	255	1	0	1	0	1
5	255	255	255	255	255	0	0	0	0	0
6	192	192	192	192	192	128	128	128	128	128
7	192	192	192	192	192	128	128	128	128	128
8	192	192	192	192	192	128	128	128	128	128
9	192	192	192	192	192	128	128	128	128	128
10	192	192	192	192	192	128	128	128	128	128

Orijinal resmin piksel değerleri ASCII karşılıklarını gösteren Tablo 2.5 ve içerisine metin gizlenmiş resmin piksel değerlerinin ASCII karşılıklarını gösteren Tablo 2.6 karşılaştırıldığında çok az pikselde değişikliğin meydana görülmektedir



Şekil 2.11: Orjinal resmin her piksellinin en az önemli bitlerine metin gizlendikten sonra yeni resim

Orjinal resmin en az önemli bitlerine metin gizlendikten sonra yeni resim için Şekil 2.11'de görüldüğü gibi gözle görülür bir renk değişimi olmamıştır.

Bu yöntemde dikkat edilmesi gereken bazı durumlar vardır. Resmin içerisine gizlenecek metinde bulunan her bir karakter için yukarıdaki örnekte de görüldüğü gibi resim içerisinden 8 piksel harcanmaktadır. Dolayısıyla içerisine bilgi saylanacak resmin toplam boyu, saklanacak metnin toplam boyu (noktalama ve boşluklarda dahil)'nun minimum 8 katı olmak zorundadır. Bilgi saklama esnasında, saklanacak bilginin verilerinin kaybolmaması için bu minimum sınıra uyulması gerekmektedir.

Herhangi bir resim üzerine gizlenen doküman, resim boyutunu değiştirmemektedir. Bu da beklenen bir sonuçtur. Boyut değişikliğinin olmaması şifreleme işleminin başarısının ayrı bir göstergesidir. Gizlenen verinin tekrar elde edilmesi için yine geliştirdiğimiz yazılım ile işlemi tersine gerçekleştirmek ve LSB bitlerini okuyarak dokümanı geri elde etmek mümkündür.

Bilginin hızla çoğaldığı ve güvenliğin her geçen gün daha da önem kazandığı günümüzde, dokümanların güvenli bir şekilde gönderilebilmesine yönelik hazırlanmış olan yeni yaklaşım başarıyla sonuçlandırılmıştır. Bu yaklaşım hazır

programlar haline getirildiği için kullanıcılar tarafından kolayca ve güvenli bir şekilde kullanılabilir. Bu yaklaşımın tek bir dezavantajı gönderilecek mesajın uzunluğunun resim boyutuna bağlı olmasıdır. Uzun mesajlar için yüksek çözünürlükte ve boyutta resimlerin kullanılması gerekir.

LSB yöntemi kullanıldığında; 55–62. bayt'lar gizlenmek istenen verinin boyutu saklanmaktadır. Gizlenmek istenen veri ise 63. bayt'dan itibaren resim içerisine gizlenmeye başlanmıştır. Burada 62 baytlık bir kayıp söz konusudur. Resmin boyutu ile mesajın boyutu arasıdaki ilişki aşağıda verilmiştir:

Resim Boyutu >=(Mesaj Boyutu x8+62) bayt olmalıdır. (2.5)

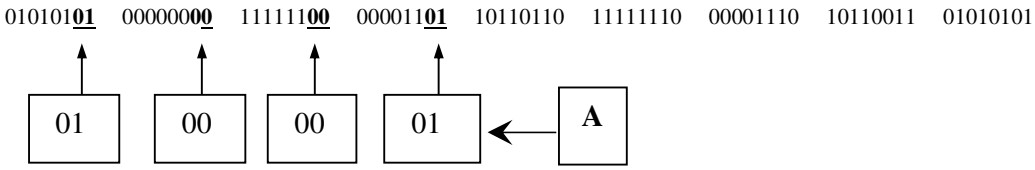
Bu aşamaya kadar LSB yönteminden bahsedilmiş ve her pikselin son biti değiştirilmiştir. Bunun yanında her piksele iki bit yerleştirmesi çalışması denendiğinde de resimde gözle görülür bir değişikliğin oluşmadığı gözlenmiştir. Bu yöntem LSB'ye göre iki kat daha avantajlıdır. Saklanmak istenen her bayt için gereken piksel sayısı yarıya düşmektedir. LSB yönteminde bir bayt veri saklamak için 8 baytlık veri alanı gerekirken İki Bit yönteminde sadece 4 baytlık bir veri alanı yeterli olmaktadır. Bu yönüyle, LSB'nin sınırlı alan sorununa bir çözüm olarak sunulabilir. Resimde bozulma LSB yöntemine göre daha fazladır. Fakat bozulmayı göz algılayamadığından bu durum göz ardı edilebilir.

Tekrar 192 sayısını ele alındığında 192 sayısının son iki biti 00'dır. Bu değer 01 yapıldığında sayının değeri 193, "11" yapıldığında sayının değeri 195 olacaktır. Bu iki fark da çok büyük farklar değildir. Bir resim için bu iki durum da düşünüldüğünde gözün bu farkları algılaması imkansızdır.

İki Bit yöntemi kullanıldığında 55–58. bayt'lar gizlenmek istenen verinin boyutu saklanmaktadır. Gizlenmek istenen veri ise 59. bayt'dan itibaren resim içerisine gizlenmeye başlanmıştır. Burada 58 baytlık bir kayıp söz konusudur. Resmin boyutu ile mesajın boyutu arasıdaki ilişki aşağıda verilmiştir:

Resim Boyutu >=(Mesaj Boyutu x4+58) bayt olmalıdır. (2.6)

A harfi için bu gömme işleminin yapıldığı dört pikselin son bitlerinde meydana gelen değişimler Şekil 2.12'de görülmektedir.



Şekil 2.12: Dört piksel içerisine A harfinin gömülmesi

2.4. Bir Piksel İçerisine Bir ASCII Kodunun Gömülmesi

Burada birinci öncelikli olarak amaçlanan resim üzerinde çok büyük deformasyona yol açmadan en büyük miktarda bilginin resim içerisine gömülmesidir. Hedefe göre her bir piksele bir ASCII kodu gömülebilmekte ve de sonuçta 1 Bayt bilgi saklanmaktadır.

(310 × 220) piksel yani (10,94 × 7,76) cm gibi çok küçük bir alana sahip bulunan resim üzerine 310 × 220 = 68200 Bayt bilgi gömülebilmektedir. Bu da (1024 Bayt = 1 KBayt) yaklaşık 66,6 Kbayt'lık bir bilginin gömülebileceği anlamına gelir [19]. Bu kadar küçük bir alanda elde edilen maksimum performans açıkça görülmektedir.

Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemi kullanıldığında 55. bayt gizlenmek istenen verinin boyutu saklanmaktadır. Gizlenmek istenen veri ise 56. bayt'dan itibaren resim içerisine gizlenmeye başlanmıştır. Her piksele (RGB) bir bayt saklanmaktadır. Burada 55 baytlık bir kayıp söz konusudur. Resmin boyutu ile mesajın boyutu arasıdaki ilişki aşağıda verilmiştir:

Resim Boyutu >=(Mesaj Boyutu x3+55) bayt olmalıdır. (2.7)

RGB : (76, 34, 212) olarak üç ana renk oranına sahip olan bir piksel (24 bit) için R=76, G=34, B=212 değerlerini alırlar ve her biri 8 bit ile ifade edilir.Şekil 2.13'de RGB (76, 34, 212) ağırlıkların 8'er bit olarak dağılımı gösterilmiştir.

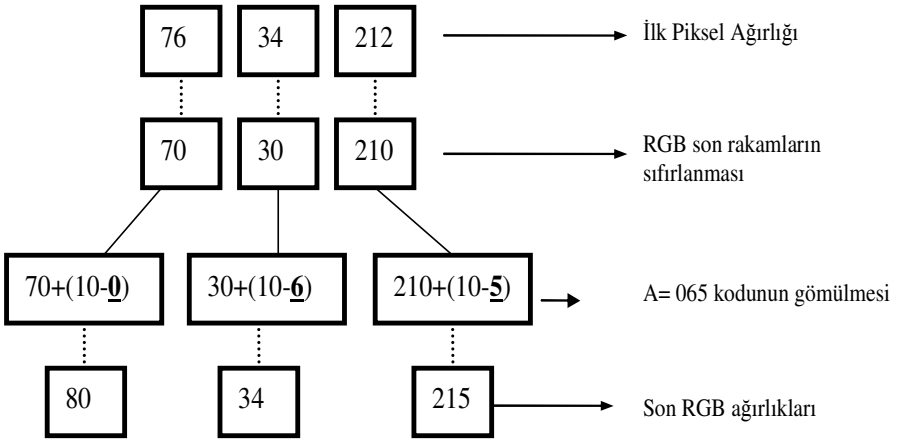
76							
R							
R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	1	1	0	0

34							
G							
G_7	G_6	G_5	G_4	G_3	G_2	G_1	G_0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	1	0	1

212							
B							
B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1	0	1	0	0

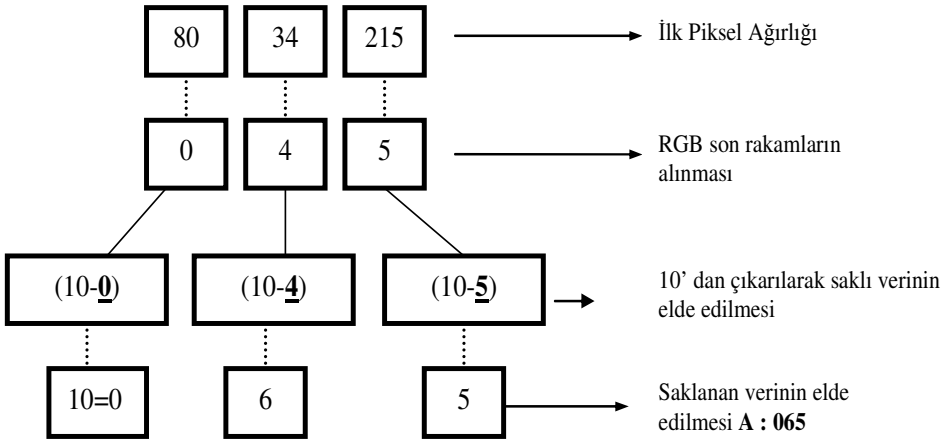
Şekil 2.13: RGB ağırlıkların 8'er bit olarak dağılımı

RGB ağırlığı (76, 34, 212) olan bir piksel içersine A harfinin ASCII karşılığı olan 065: (01000001) değerinin saklanması Şekil 2.14'de gösterilmektedir.



Şekil 2. 14: Bir piksel içerisine bir ASCII kodunun gömülmesi

RGB ağırlığı (76, 34, 212) olan bir piksel içersine A harfinin ASCII karşılığı olan 065: (01000001) değerinin yeniden elde edilmesi Şekil 2.15'de'de gösterilmektedir.

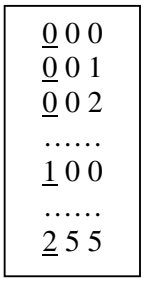


Şekil 2.15: Piksel içerisine bir ASCII kodunun çıkarılması

.

2.5. "R" Kodlama Ağırlığının Değiştirilmesi

Bu yaklaşımda önemli olan orijinal piksel ağırlıklarının olabildiğince düşük oranlarda değişikliğe uğramasıdır. Aynı oranda orijinal resim ile kodlanmış resim arasındaki benzerlik de artacaktır. Buradan hareketle "R" ağırlığının kodlamasını diğer iki ağırlığın (G–B) kodlanmasından ayırmak gerekir. Bunun nedeni "R" ağırlığına gömülen ASCII kodunun ilk biti MSB olup değer olarak 0, 1 ve 2 değerlerini almaktadır.. Bu durum Şekil 2.16'da ilgili bitlerin altı çizilerek örneklenmiştir.



Şekil 2.16: Herhangi bir ASCII kodunun ilk biti (MSB) olup değer olarak 0, 1 ve 2 değerlerini aldığının gösterimi

2.6. Kısmi Optimizasyon Yöntemi İle Veri Gömme Yöntemi

Resim 8 farklı bölgeye ayrılarak ilk önce 1.bölgeyi oluşturan piksellerin RGB ağırlıkları belirlenir. Eş zamanlı olarak gömülecek dosyanın ASCII kodları da belirlenmektedir. Bu bölgeye 8–farklı optimizasyon uygulanarak sonuçlar birbirleri ile karşılaştırılır. Orijinal piksellerle karşılaştırıldığında en az hata/bit oranına sahip olan Optimizasyon tespit edilerek bir değişkende saklanır. Her bölge için bu işlemler tekrarlandığında, sonuçta tüm bölgelerin hangi optimizasyonla gömme işlemine tabi tutulduğunda daha az bozulma oluşacağı tespit edilir.

Tüm resim bölgeleri kaydedilen bu değişkenlere göre 8–bit ASCII kodlarını gömmek üzere düzenlenir. Bu işlemler gizli dosyanın son ASCII koduna ulaşılıncaya kadar sürdürülür. Gizli dosyanın tamamı gömüldüğünde kullanıcı şifresi tespit edilir. Son olarak içerisinde gizli veri gömülü olan resim tüm pikselleri ile dosyaya yazılarak gömme işlemi sonlandırılır [19].

2.7. Maskeleme ve Filtreleme

Görünür damgalama uygulamalarında kullanılan yöntemlerden biri Maskeleme ve filtreleme'dir.. Gri tonlamalı resimler ve 24 bit BMP formatı resimlerde kullanılır. Resimde gözle görülür işaretlemeler yapılır.

Şekil 4.17'de bir uygulama görülmektedir. Yapılan işlem ile belirli piksellerin parlaklık değerini %15 arttırılmış ve "Invisible Man" ©1997, Neil F. Johnson metni resme eklenmiştir.

Şekil 2.17: Maskeleme ve filtreleme uygulaması

2.8. Logaritma Kullanılarak Rasgele LSB ekleme

Sıralı LSB yönteminde verinin resmin satırlarına ya da sütunlarına sıra ile yerleştirilmesinden dolayı gizli mesajın elde edilmesi işlemi oldukça kolaydır. Bu yüzden verileri rasgele bir şekilde resmin içine saklamak daha güvenlidir.

Ayrık logaritma fonksiyonu kullanarak veri gizleme M. M. Amin, M. Salleh, S. Ibrahim, ve M. R. Katmin tarafından geliştirilmiş bir yöntemdir [20].

$Y_i=a^i \bmod p$ şeklinde tanımlanan ayrık logaritma fonksiyonu resim içine rasgele şekilde veri gizlemeyi sağlar. Burada Y_i , mesajın i . bitinin resmin içinde saklanacağı pozisyonu; i gizlenecek mesajın bit indeksini göstermektedir. Fonksiyondaki p büyük bir asal sayı ve a ise p 'den üretilen asal bir köktür. Yani p ile a kendi aralarında asal olmalıdır. P değerinin asal olmasının nedeni aynı değerin

tekrar üretilmemesinin sağlanmasıdır. Gizlenecek metnin uzunluğu m , içine veri gizlenecek resmin büyüklüğü l ise p degeri, m< p< l şartını sağlamalıdır [16].

Rasgele bir yerleşim sağlamak amacıyla ayrık logaritma fonksiyonu kullanılmıştır. Verinin rasgele gizlenmesi sıralı LSB yöntemine göre daha güvenlidir. Gizlenmiş verinin saldırgan tarafından elde edilmesi bu yöntemde daha zordur. LSB yönteminin özelliği nedeniyle de steganografi uygulanan resmin boyutunda bir değişiklik olmamaktadır. Son bite ekleme yönteminin dezavantajı gönderilecek mesajın veya dokümanın uzunluğunun resim boyutuna bağlı olmasıdır. Gizlenecek veri miktarını arttırmak için çeşitli sıkıştırma algoritmaları da kullanılabilir.

2.9. Veri Gizleme Amacıyla Kullanılan Güncel Programlar

2.9.1. Camera/shy

Programı hazırlayan Hactivismo olarak adlandırılan grup Camera/Shy adını verdikleri bu ücretsiz yazılım ile kullanıcılarının fotograflar içine gizlenmiş mesajları okuyabileceklerini ve böylece bilinen polis izleme metotlarını aşacaklarını belirtiyorlar.

2.9.2. Türksteg

Erciyes Üniversitesi Bilgisayar Mühendisliği Bölümü'nce geliştirilen Türkiye'nin ilk Türkçe veri gizleme programı "TürkSteg" sayesinde, internet üzerinden gönderilen gizli veriler, bir resmin içerisine saklanabilmektedir.

TürkSteg sayesinde bir resmin içerisine gizlenen veriler, anlaşılamamakta, resmin boyutlarında herhangi değişiklik de olmamaktadır. Gizlenen verinin veya dokümanın farklı kişiler tarafından elde edilmesi mümkün olsa da resim içerisindeki bilgiyi elde etmeleri oldukça zordur. Veriler veya dokümanlar, seçilen bir resim içerisine otomatik olarak gizlenmektedir. İçerisine bilgi gizlenen resim karşı alıcıya gönderilmekte ve bu resmi alan kişi yine aynı yazılımı kullanarak resim içerisinden bilgileri otomatik olarak elde edebilmektedir. Bu güvenli haberleşmeyi yapmak

isteyen taraflarda 'TürkSteg' yazılımının bulunması gereklidir. Bu yazılım otomatik olarak hem gizleme hem de gizli mesajları tekrar geri elde etme kabiliyetine sahiptir.

2.9.3. Photo watermark 1.0

Program her türlü yazı biçimini ve tüm yaygın resim biçimlerini desteklemektedir. PNG, JPEG, BMP, GIF gibi daha birçok resim biçimi ile çalışabilmektedir.

Program, küçük boyutlu olduğu gibi aynı zamanda kuruluma da ihtiyaç duymaz. Tek bir tıklama ile hiçbir teknik konu bilmeden resimlere yazı eklenebilir. Yazı eklenen resmin boyutu ayarlayabilir.

2.9.4. Eyemage IIE

Kurulum gerektirmeyen yazılım, birkaç basamakta istenilen BMP uzantılı resim içine veri gizlememize olanak sağlamaktadır.

Yazılım, verileri iki türlü şifreleme yöntemi ile korumaktadır:

1) Direkt olarak kullanıcının belirlediği parola ile güvenlik sağlanmaktadır. En az dört karakter girerek oluşturulan kod olmaksızın veriler resimden alınamamaktadır.

2) Bir diğer koruma yöntemi de soru temellidir. "Beş farklı basamakta, on iki sorudan oluşan bir algoritmaya, her basamakta belirtilen cevap verilmeksizin çözülmeyen bir koruma" şeklinde tarif edilen bu yöntem, belirlenen sorulara doğru cevap verilmeksizin verilerin görülmesini engellemektedir. Böylece istenilen cevaplar olmaksızın verinin resimden alınması ölçülemeyecek oranlarda zor bir olasılığa düşmektedir.

2.9.5. Watermark Factory

Bu program ile internette yayınlamak ya da bastırmak istenilen fotoğrafların uygun görülen bir yerine kişiye ait olduğunu gösteren bir işaret ya da yazılar eklenebilir. Belirlenen işaretin (resim ya da logo) ya da yazının saydamlığını büyüklüğünü ve rengini ve fotoğrafların neresinde duracağını ayarlanabilmektedir. Ayrıca dosya ve tarih bilgisini ekleme gibi seçenekleri de mevcuttur.

2.9.6. Visual watermark

Visual Watermark resimleri korumak ve çalınmasını önlemek için geliştirilmiş bir programdır. Bu programla resimlerin üzerine kullanıcının belirleyebileceği bir işaret koyulabilmektedir.

3. RESİM FORMATLARI

3.1. Giriş

Bir piksellik görüntülü renk paletinde kaç renk olduğu tamamen görüntünün derinlik adı verilen özelliği ile ilgilidir. Bir piksel için bir bit derinlik ayrılmışsa, bu piksel ya 1 ya da 0 değerini alabilir. Böyle bir pikselin alabileceği renklerin sayısı 2'dir. 8 bit derinliğinde olan bir görüntü 2^8 =256 renk ile sınırlıdır. Böyle bir durumda 1 piksel için tahsis edilmiş olan bellek alanı 1 bayt olacaktır. 2x2'lik ve 8 bit derinliğindeki bir görüntü 4 bayt, 3x2 piksel boyutunda ve 8 bit derinliğindeki bir görüntü 6 bayt belleğe gereksinim duyar. Bir piksel için tahsis edilmiş 8 bit farklı biçimlerde kullanılabilir. Örneğin siyahtan beyaza kadar bir skalayı temsil ediyorsa 8 bitlik 256 ton gri skala bir görüntü elde edilmiş olur. Gerçeğe yakın görünmesine rağmen bazı renkler ölçekte bulunan en yakın renk ile değiştirilmiştir. Böyle bir kayıp tahsis edilen bit sayısının azlığından doğmuş olup, her piksele daha fazla bellek ayrılarak aşılabilmektedir.

Renklerin gerçek yaşamda olduğu gibi görünmesi için Yeşil Kırmızı ve Mavi renklere birer bayt (8 bit) ayırmak gerekir. Böyle bir görüntünün derinliği (3 bayt) 24 bittir. 24 bitlik görüntünün temsil edebildiği renk sayısı 2^{24} yani 16.777.216 (16 Milyon) renktir. Böyle bir görüntüde 2x2 piksel boyundaki bir görüntü 12 bayt bellek alanına gereksinim duyar. 2x3 boyutundaki bir görüntü 18 bayt bellek alanı gerektirir.

Matbaalar için gereken CMYK görüntülerde RGB sistemi kullanılmadığından temsil edilen renk sayısı daha az olmasına rağmen piksel başına gereken bellek 4 bayttır [21]. Alfa kanalı adı verilen ve genelde maskeleme amaçlı kullanılan her ilave kanal için bellekte piksel başına bir bayt ilave etmek gerekir. Layer adı verilen ve RGB görüntülerin tamamını kapsayan katmanların her biri için piksel başına 3 bayt bellek hesaplamak gerekir.

1 bit = 1/8 bayt =2^2 renk = 2 renk, Siyah ve Beyaz (3.1)

4 bit = 1/2 bayt =2^4 renk = 16 renk, Ölçekli renk (indexed) (3.2)

8 bit = 1 bayt =2^8 renk = 256 renk, Ölçekli renk (indexed) (3.3)

16 bit = 2 bayt =2^{16} renk = 65536 renk, çoklu renk (3.4)

24 bit =3 bayt =2^{24} renk=16.777.216 renk RGB gerçek renk (3.5)

+8 bit = +1 bayt = +2^8 = İlave her kanal (Channel) (3.6)

+24 bit = +3 bayt = +2^{24} renk = İlave her katman (Layer) (3.7)

3.2. Ekran Kartı ve Piksel Derinliği

Renklerin gerçek değerleri ile ekranda temsil edilebilmeleri için ekran kartı içerisinde bulunan ekran belleğine sığmaları gerekmektedir. Örneğin 640x480 piksellik bir ekran görüntüsü 16 renk olarak kullanıldığında temsil ettiği 16 renk dışında renkleri en yakın renge çevirir. Böyle bir ekran modu için: 640x480x(1/2 bayt)= 153600 bayt gerekir. Ekran 256 renk olarak kullanıldığında 640x480x1= 307200 bayt gereklidir. Bu renk derinlikleri bellek kapasitesi 500K olan ekran kartları ile karşılanabilmektedir. Aynı çözünürlükte gerçek renkleri görebilmek için 640x480x3 bayt= 921600 (900K) gerekir. Böyle bir çözünürlüğün 500K belleği olan bir ekran kartı ile karşılamak mümkün değildir ve 640x480 boyutlarındaki ekranda gerçek renk görebilmek için tek yol 1MB belleği olan bir kart edinmektir.

Günümüzde kullanılan sıkıştırma yöntemleri kayıplı ve kayıpsız olmak üzere ikiye ayrılır. Her ikisi de ileri derecede büyük dosyaların boyutunu düşürür. Kayıpsız yöntemlere örnek olarak Wave Table ve LZW verilebilir. Kayıplı sıkıştırmalar kayıt sırasında veri kaybı ile birliktedirler. Bu kayıplar daha sonra karşımıza çıkabilmektedirler. Kayıplı kayıtlara örnek olarak JPEG gösterilebilir.

Görüntü dosyaları iki şekilde depolanabilir:

1. Eğriler, alanlar ve dolduruldukları renkler olarak
2. Noktalar topluluğu olarak.

İlkine vektör tabanlı, ikincisine piksel tabanlı görüntü denmektedir. Vektör tabanlı görüntülerin avantajı istenilen ölçülere büyütme yapılabilmesidir. Piksel tabanlı

görüntülerin dosya boyutu büyütme oranı arttıkça artmaktadır. Vektör grafikleri genelde rahatça piksel tabanlı grafiklere dönüştürülebilmektedir. Tersi ise her zaman kolay değildir. Vektör grafikleri içerisinde pikselli grafik kullanmak mümkündür ancak tersi imkansızdır.

Çoğu zaman yanlış olarak kullanılsa bile çözünürlük birim içerisinde nokta sayısıdır. Görüntüde birim olarak piksel per inch veya piksel per cm kullanılması yaygındır. 10 cm uzunluğunda ve 20 cm enindeki bir görüntü santimetrede 100 piksel çözünürlüğe sahip ise boyunda 10x100=1000 piksel, eninde ise 20x100= 2000 piksel bulunur. Görüntüdeki nokta sayısı enxboy =2000x1000 = 2.000.000'dur. Baskı cihazlarının çözünürlüğü ise dpi (dot per inch) yani bir inch başına vurduğu nokta sayısı ile ölçülür. Ortalama olarak görüntü çözünürlüğü baskı cihazının sahip olduğu çözünürlüğünün 1/4 kadarı olabilir. Örneğin nihai çalışma 300 dpi çözünürlüğe sahip bir printer ile basılacaksa inch başına 75 piksel (300/4) görüntü çözünürlüğü yeterlidir. Daha düşük çözünürlüklerde kalite kaybı, daha yüksek çözünürlüklerde gereksiz dosya boyutu artışları izlenmektedir.

3.3. BMP (Bitmap)

BMP Windows ve Microsoft'un PCX formatını değiştirerek geliştirdiği bir formattır. Windows 3.1 ve 95 ile birlikte gelen Paint programı görüntüleri bu formatta işler. OS/2 ile birlikte gelen Paint programının BMP dosyaları çok az bir farklılık gösterir. BMP formatı 1–24 bit arasında değişen bir piksel derinliğini içerebilir. Sıkıştırma seçeneği başlangıçta bulunmamakta idi. Sonradan RLE yani Run–Lenght–Encoding sıkıştırma yöntemi BMP dosyaları için benimsendi. Opsiyon olan bu sıkıştırma görüntüde detay kaybına yol açmaz, yani kayıpsız sıkıştırma yöntemlerindendir. BMP formatı alıcı bilgisayarında Paint'den başka görüntü programı bulunmadığı durumlarda kullanılır. Gidecek resim OS/2 işletim sistemine gidecekse seçeneklerde OS/2 BMP'si işaretlenir.

Bitmap, bilgisayar ekranında (display space) bir resmi görüntülemek için gereken tüm renk bilgilerinin ekranı oluşturan hex piksel (ekran çözünürlülüğü dahilinde, adreslenebilir en küçük resim parçası) için renk bilgilerinin verildiği resim

formatlarına verilen genel addır. GIF, JPEG, BMP, PCX örnek Bitmap formatlarıdır. Bitmap dosyaları ekranda görebilmek ve yazıcılardan bastırabilmek için özel bazı programlar gerekir (Photo Shop, XView, ACDSee vb gibi).

Bitmap formatları, resmin çözünürlülüğüne bağlı sabit bir tanımlama ile resmi oluştururlar (Raster Yöntemi). Bu haliyle, eğer resmi ölçeklendirilirse (küçültme/büyütme) birçok ayrıntı kaybedilebilinir. Vektör tabanlı grafik formatlarında ise, tanımlamalar bağıl olduğu için ölçeklendirmelerde kayıp önlenir (CAD formatları, Post Script (PS, EPS) formatları gibi).

Çalışmamızda da kullanılan, en temel resim formatı BMP'dir. Aslında BMP'nin birbirinden farklı bir kaç türü vardır. Özellikle bir X–Windows kullanıcısı ile MS–Windows ya da OS/2 kullanıcısı için farklar mevcuttur.

X–Windows üzerindeki BMP formatı, sadece 2 rengi desteklemektedir. MS–Windows ya da OS/2 üzerindeki BMP formatının X–Windows'daki karşılığı XPM'dir (pixmap). MS–Windows üzerinde BMP 16 ya da daha çok renk kaydedebileceğiniz, herhangi bir sıkıştırma yapmayan oldukça hızlı bir formattır. Bu formatta resmin içindeki renk sayısı değil, resmin büyüklüğü önemlidir.

16 renk, 800x600 çözünürlüğünde bir BMP dosyası, 800x600x1/2=240000 bayt yer kaplamaktadır. Resmin içinde 1, 2 ya da 12 renk olması hiç önemli değildir. 256 renk olarak kaydedilen bir dosya ise, 800x600x1=480000 bayt yer tutmaktadır. (256 için 8 bit=1 bayt gerekli).

3.4. GIF (Graphics Interchange Format)

BMP, çok hızlı bir format olmasına karşın, oldukça fazla yer kapladığı için pek tercih edilmemektedir. Kullanılan dosya 256 renkten fazlasını içermiyorsa, GIF iyi bir çözüm olabilir.
GIF, Compuserve'nin geliştirdiği bir resim formatıdır. İyi bir sıkıştırma algoritması vardır ve görüntüleme de oldukça hızlı bir şekilde gerçekleştirilmektedir. 256 renk

dışında (8 bit) herhangi önemli bir sorunu yoktur. Bunun yanında GIF, Web browserlar ile görüntülenen resimler için standart bir resim formatıdır.

GIF formatının iki farklı versiyonu vardır: 87a ve 89a. 89a versiyonu, tek bir GIF dosya içinde birden çok GIF formatlı resim yerleştirilmesine ve anime edilmesine olanak tanır (animated gif). Ayrıca, GIF89a versiyonu, "interlaced" (katmanlı) görüntü saklama özelliğine de sahiptir. Bu, özellikle internet üzerindeki resimlerde kullanılır. Böylece, kullanıcı, GIF formatındaki resmi, her seferinde 1 katman gelecek şekilde ekranında görür ve resmin bütünü hakkında, tüm resim gelmese bile, fikri olur.

CompuServe firmasının Graphics Interchange Format (GIF) dosyaları internet üzerinde oldukça yaygın kullanılan bir formattır. Az sayıda renk içeren (1 ile 8 bitlik) dokümanlarda oldukça iyi sıkıştırma sağlaması, animasyonlarda zamanlama ve farklı boyutlardaki resimleri bir arada tutma desteği, saydam renk tanımlanması bu format'ı popüler yapan nedenlerden sadece bir kaçıdır. Ancak Photoshop gibi resim işleme programlarının çoğu GIF formatının tüm özelliklerini kullanamamaktadır. Bu nedenle bu format ile çalışırken sıklıkla başka programlara gereksinim duyulmaktadır. Telefon hatları üzerinden hızlı iletiyi sağlamak için LZW sıkıştırma yöntemini kullanmaktadır. GIF dosyaları bitmap, gri skala ve indekslenmiş renk sisteminde olabilmektedir. Gerçek renk desteği yoktur. GIF resimleri sıralı veya sırasız kaydedilebilmektedir.

Ayrıca dosya ile birlikte metin kaydedilebilmektedir. Sıralı GIF dosyaları yükleme esnasında satır satır gelerek resim bitiminden önce neye benzeyeceğine dair bir ipucu verirler. Saydamlık tanımlanması için GIF89a Export komutu kullanılarak saydam olacak renk belirlenebilir.

3.5. PNG (Portable Network Graphics)

PNG, "Taşınabilir Ağ Grafiği" anlamındaki (Portable Network Graphics)'in kısaltmasıdır ve kayıpsız sıkıştırarak görüntü saklamak için kullanılan bir saklama

biçimidir. PNG biçiminde paletli ya da gerçek renkte görüntüler seçimlik bir saydamlık kanalıyla saklanabilir.

Web için tasarlanmış yeni bir formattır. GIF gibi 256 renkle sınırlı olmayan, JPEG gibi sıkıştırma oranı arttıkça kalite kayıpları oluşmayan bu format, aynı zamanda kademeli olarak görüntünün oluşması özelliğine de sahiptir.

PNG, kayıpsız bir filtreleme ile tüm renk bilgileri ve tüm alfa kanallarını koruyarak sıkıştırmayı gerçekleştirdiği için, eski sürüm web tarayıcıları tarafından desteklenmeyecek olmasına rağmen geleceğin formatı olarak görülmektedir. GIF 256 renkten fazlasını desteklememekte, JPEG ise alfa kanallarına izin vermemektedir.

3.6. JPEG (Joint Photographics Experts Group)

Çok renkle (256 renkten fazla) çalışıldığında, GIF formatını kullanması mümkün değildir. BMP olarak saklandığında çok fazla disk alanı kaplayabilir. Onun yerine JPEG daha iyi bir alternatif olabilir. Yalnız, JPEG az renk içeren uygulamalarda hem kaliteyi düşürmekte hem de dosya boyutunda önemli bir değişiklik sağlamamaktadır.

Standart JPG formatında, resmin kalitesinden bir miktar ödün vererek sıkıştırma uygulanır. Böylece dosya boyu bir hayli düşer. Özellikle 24 bit true color uygulamalarda resim kalitesinin düştüğünü anlamak mümkün değildir. Bu tip uygulamalarda JPG tercih edilir.

JPEG'den ne kadar sıkıştırma istendiği faktör miktarı (0–100 arası bir faktör) seçilir ama genellikle 5–95 arası kullanılır. 95'den fazlası detay kaybına yol açar, 5'den küçüğü de dosyayı fazla küçültmez.

Bir de 24 bit–8 bit çevrimli JPG formatı vardır. JPG de, GIF gibi, Web listeleyiciler tarafından görüntülenebilen standart bir formattır. JPG, ISO standardı ile tanımlanmış bir formattır ve birçok değişik kodlama sistemleri içerir.

JPEG formatı sık kullanılan bir formattır. JPEG veya JPG formatının özelliği gerçek renk değerlerini içermesidir. Bu nedenle fotoğrafik yani grafiksel olmayan

görüntülerin gösterilmesinde GIF formatına göre üstündür. JPEG sıkıştırma yöntemi görüntünün algılanması için çok önemli olmayan detayları etkili bir şekilde bulup atan ve dosyayı bu şekilde sıkıştıran bir format olduğundan kayıplı formatlar arasında sıralanır. Yok edilen detay miktarı ve sıkıştırma oranı arasında orantı olduğundan bu dengeyi iyi korumak gerekmektedir. Daha fazla sıkıştırma daha fazla detay kaybı, daha az sıkıştırma daha büyük dosya demektir. Bu dengeyi en iyi şekilde değerlendirecek olan insan gözüdür. Bu nedenle bir dosyanın kopyası JPG olarak kaydedildikten sonra açılıp tekrar değerlendirilmelidir. Kaybedilen detayların geri getirilmesi söz konusu olmadığından dosyanın bir kopyasını kayıpsız bir yöntem ile korumakta fayda vardır. Her kaydediliş sırasında kayıp miktarı arttığından JPG dosyaları sadece nihai işlerin oluşturulması için kullanılır. Ara kademelerde kullanılmaları uygun değildir. Maksimum kalitesi göz tarafından orijinalin aynısı gibi görünmesine rağmen yine kayıplar olabilecektir.

3.7. TIFF (Tagged Image File Format)

1, 8, 24 bitlik formatları vardır. Hepsinin sıkıştırılmış ve sıkıştırılmamış 2 farklı tipi mevcuttur. 1 bit olanı fakslarda dosya iletimi için kullanılır. Çok renkle uğraştığınızda, zaman önemli, yer çok önemli değilse, TIFF'i kullanmanızda fayda vardır. JPEG'e göre daha az küçülme sağlasa da, hızı ile bu açığını kapatmaktadır.

TIFF formatı farklı işletim sistemleri ve uygulamalar arasında kayıpsız ve esnek bir dosya değiş tokuşunu sağlaması nedeniyle tüm çalışmalar için uygun bir format olarak bilinmektedir. TIFF'in desteklediği birçok sıkıştırma vardır. Bunlar arasında en çok kullanılan kayıpsız LZW sıkıştırma yöntemidir. TIFF ayrıca çok sayıda alfa kanalını desteklemektedir. Kayıt sırasında fotoğrafın kullanılacağı işletim sistemi olarak PC veya Mac seçilebilmektedir. TIFF dosyaları ikili dosya, indekslenmiş renk, gerçek renk RGB, CMYK, Lab gibi neredeyse tüm biçimlerini destekler. TIFF dosyalarında katman (Layer) desteği bulunmaz.

3.8. Photoshop EPS

Encapsulated PostScript (EPS) lisanı ile yazılan bir dosya biçimidir. Vektör ve Piksel tabanlı görüntüleri alabilmektedir. Patenti Adobe firmasında olan bu formatı daha çok matbaalar kullanmaktadır.

3.9. Raw

RAW değişik bilgisayarlar ve işletim sistemleri arasında bilgi iletimine izin veren esnek bir formattır. Kanal sayısı, her kanaldaki piksel derinliği dosya uzantısı ve başlık bilgileri tanımlanabilir. Kayıt sırasındaki parametre bilgileri açmak amacıyla dosyayı alan kişiye verildiğinde RAW dosyaları kolaylıkla açılabilmektedir.

3.10. Scitex CT

Scitex Continuous Tone (CT) formatı Scitex bilgisayarları RGB, CMYK ve Gri skala resimler tarafından kullanılan bir formattır. Patentli Scitex baskı sistemleri ile basılan resimlerde ve elde edilen filmlerde Moire desenlerine çok az rastlanmaktadır. Bu nedenle yüksek kaliteyi isteyen profesyonel birçok dergide kullanılmaktadır.

3.11. XCF (Experimental computing facility)

XCF Experimental computing Facility adlı çalışma grubunun öncülüğünde doğan GIMP programının formatıdır. Bu format çok sayıda alfa kanalı ve layerleri desteklediği gibi, path desteği bulunmaktadır.

3.12. Targa Truevisin

Video kartlarının yaygın olduğu zamanlarda AT&T laboratuarlarında ve DOS ortamında film ve animasyon üretmek için neredeyse oldukça yaygın olan Targa dosya biçimleri RGB gerçek renk ve tek alfa kanalını desteklemektedir.

4. SAYISAL RESİM İÇERİSİNE VERİ GİZLEME UYGULAMALARI

4.1. Giriş

Bu tez çalışmasında; LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri yardımıyla BMP formatındaki bir resim içerisine herhangi bir metin dosyasındaki veri gizlenerek yeni bir resim dosyası oluşturulmuştur. Gizlenmiş bilgiyi içeren resim dosyası kablosuz ortamdan TCP ve UDP protokollerini kullanarak transfer edilmiş ve daha sonrasında gizlenmiş bilgiyi içeren resim dosyasından, iletilmesi istenilen bilgi alınmıştır.

4.2. LSB Yöntemi

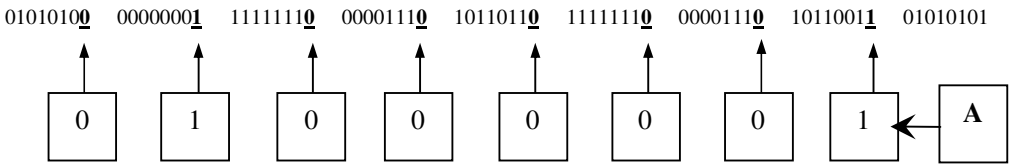
Karakteristik olarak tam–renk (full–color) bir resim 24bit/piksel hücrelerden oluşmakta olup her bir renk bileşeni 8 bittir. Gri ölçekli bir resimde ise bu oran 8–bit/piksel olarak kullanılmaktadır. Bir dijital resim; renk ve yoğunluk değerlerinin bir matris dağılımından oluşmaktadır.

En basit stenografik gömme yöntemi, mesajları en düşük değerlikli bit içerisine rasgele olmayan bir sıra ile resim içerisine gömmektir. Bu yöntem LBS yöntemidir. LSB uygulaması sonucunda oluşan çok küçük değişim ve bozulmalar insan gözünün algılayabileceği seviyede değildir.

Diğer yöntemlerde mesaj; yalancı–rasgele gürültülerle işlenerek gizli örtülü resim içerisine bu işlem esnasında ya da daha önce eklenmektedir. LSB gömme yönteminin avantajı, basitliği sayesinde birçok teknik tarafından kullanılmasıdır [22]. Bununla birlikte sağlamlık ve güvenlik konuları göz önüne alındığında birçok zayıf noktaları mevcuttur. LSB yöntemi herhangi bir filtreleme veya değişikliğe karşı aşırı derecede hassastır. Ölçeklendirme, döndürme, kesme, gürültünün ilave edilmesi veya kayıplı sıkıştırma gibi durumlarda orijinal mesaj zarar görmesi olasıdır. [23]. Bundan başka dışarıdan yapılacak bir saldırı ile LSB bitlerinin tümüyle sıfırlanması ile tüm mesaj

kolaylıkla silinebilir. Bu durumda da resim üzerinde algılama kalitesinin çok küçük olduğu minimum değişiklikler oluşmaktadır.

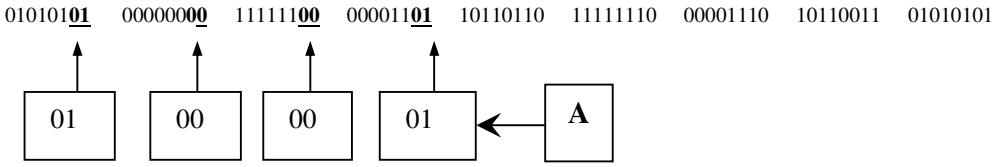
A harfi için bu gömme işleminin yapılışı ve sekiz pikselin son bitlerinde meydana gelen değişimler Şekil 4.1'de görülmektedir.



Şekil 4.1: Sekiz piksel içerisine A harfinin gömülmesi

Bu tez çalışmasında LSB yönteminin yanı sıra İki Bit yöntemi üzerinde çalışılmış ve başarılı sonuçlar elde edilmiştir. LSB yönteminde her seferinde bir bit üzerinde çalışılırken İki Bit yönteminde her piksele yerleştirilen bit sayısı ikidir. Bunun yanında her piksele iki bit yerleştirmesi çalışması denendiğinde de resimde gözle görülür bir değişikliğin oluşmadığı gözlenmiştir. Saklanmak istenen her bayt için gereken piksel sayısı yarıya düşmesi bakımından bu yöntem LSB'ye göre iki kat daha avantajlıdır. LSB yönteminde bir bayt veri saklamak için 8 baytlık veri alanı gerekirken İki Bit yönteminde sadece 4 baytlık bir veri alanı gerekmektedir. Bu yönüyle, LSB'nin sınırlı alan sorununa bir çözüm olarak sunulabilir. Resimde bozulma LSB yöntemine göre daha fazladır. Fakat bozulmayı göz algılamadığından bu durum göz ardı edilebilir.

A harfi için bu gömme işleminin yapılışı ve dört pikselin son bitlerinde meydana gelen değişimler Şekil 4.2'de'de görülmektedir.



Şekil 4.2: Dört piksel içerisine A harfinin gömülmesi

Renkli resim dosyaları boyut olarak oldukça geniş yer kaplarlar, ancak hafıza alanını azaltılması ve haberleşme gereksinimlerinde dolayı bazı sıkıştırma planları

geliştirilmiştir. Bitmap ve GIF dosyaları kayıpsız sıkıştırma algoritmaları kullanırlar. Kayıpsız sıkıştırma algoritmalarında sıkıştırma işlemi tersi ile açılan resim orijinal resim ile birebir özdeştir.

JPEG dosyaları kayıplı sıkıştırma algoritmaları kullanmaktadırlar. Sıkıştırılmış resim dosyası görsel açıdan orijinaline benzemekle birlikte piksel bazlı ele alındığında birçok kayıpların meydana geldiği görülmektedir. [24,25]. Her ne kadar her iki tip de stenografide kullanılmakta ise de bilgi gizleme araçları açısından jpg sıkıştırma bitmap ve gif formatlarına göre çok daha karmaşıktır.

Bu çalışmadaki yaklaşım, sayısal resimlerin içerisine bir verinin nasıl gizlenebileceği üzerinedir. Yapılan işlemi anlamak için en az öneme sahip bit kavramını iyice kavramak gerekir. İkili sayı sistemine göre 10101011 sayısını ele alalım. Bu sayı onluk sayı sisteminde 171 sayısına karşılık gelmektedir. En sondaki bitin 1 veya 0 olması bu değeri çok fazla değiştirmeyecektir. En sondaki bit değerimiz eğer 0 olsaydı bu değer 170 olacak ve renk üzerinde gözle görülecek büyük bir değişikliğe neden olmayacaktır. Bu sondaki bite LSB denir ve fazlaca etkisi olmadığı ortadadır. Burada sunulan yaklaşımda, LSB bitleri yerine saklamak istediğimiz dokümanın bit değerleri ile sırasıyla yer değiştirilmiş ve daha sonra yerleştirilen bu bitler tekrar elde edilmiştir.

4.3. Bir Piksel İçerisine Bir ASCII Kodunun Gömülmesi

Bu çalışmada resim üzerinde çok büyük bozulmaya yol açmadan en büyük miktarda bilginin resim içerisine gömülmesi amaçlanmaktadır. Bu amaçla resmin her bir pikseline bir ASCII kodu gömülmekte ve bu şekilde her pikselde 1 bayt bilgi gizlenebilmektedir.

Bu yöntem LSB yöntemine göre veri gömme oranı bakımından daha avantajlıdır. LSB yönteminde 8 bit veri 64 (8x8) bit veri içerisine saklanabilirken bu yöntemle aynı miktarda veri sadece 24 bit veri içerisine saklanabilmektedir. Bu oranın düşmesi beraberinde resimde çok daha fazla bozulmaya yol açabilmektedir. Gözün farkı algılayamaması bakımından bu durum göz ardı edilebilir.

4.4. TCP ve UDP Protokolleri

TCP bir telefon görüşmesi ile benzetilebilir. Eğer birine telefon etmek istiyorsanız; o kişinin bir telefonu, bir telefon numarası ve gelen çağrıyı beklemesi gerekir. Kişi sizden gelen telefonu yanıtladıktan sonra siz; güvenli ve çift yönlü iletişim akışına sahip olmuş olursunuz. Eğer karşı taraf telefonu kaldırırsa iletişim başlar. Aynı anda her iki kişi de birbirleri ile konuşabilirler. TCP protokolünü kullanan ağ bağlantısında; istemci bilgisayar telefon eden kişiye, sunucu bilgisayar ise çağrıyı bekleyen kişiye benzetilebilir. istemci bilgisayar, sunucu bilgisayara bağlanma girişiminde bulunduğunda sunucu bilgisayarın; çalışıyor ve ağ üzerinde bir adrese sahip olması gerekmektedir. Bunlara ek olarak port üzerinden gelen bir bağlantıyı beklemelidir. Bir TCP bağlantısı kurulduğunda istemci ve sunucu bilgisayar çift yönlü ve güvenli veri iletim hakkına sahip olurlar. Bu iki bilgisayar bağlantı sonlandırılana ya da kopana kadar iletişime devam edebilirler.

Aktarım Denetimi Protokolü aşağıdaki hizmetleri sağlar:

- IP veri birimlerinin teslim edileceğini garanti altına alır.
- Programların gönderdiği büyük veri bloklarını parçalar ve yeniden birleştirir.
- Bölünen blokların doğru sırada ve düzenli teslimini sağlar.
- Sağlama hesapları yaparak iletilen verilerin bütünlüğünü denetler.
- Verilerin başarıyla alınıp alınmadığına bağlı olarak onaylama iletileri gönderir. Seçici bildirimler yoluyla alınmayan verilerin alınmadı bildirimini de yapar.
- İstemci/sunucu veritabanı ve e–posta programları gibi güvenilir oturuma dayalı veri iletimleri kullanmak zorunda olan programlar için tercih edilen bir taşıma yöntemi sağlar.

Kullanıcı veri birimi protokolü, uygulamalar arasında datagrams adı verilen veri paketlerinin gönderimini sağlar. TCP bir telefon görüşmesine benzetilirse, UDP de birine mektup gönderilmesine benzetilebilir. Sunucuya bağlantı kurulmaksızın, istemci sunucuya veri paketi (datagrams) gönderdiğinde datagram da mektuba benzetilebilir. İstemci ve sunucunun direk olarak birbirlerine bağlanma durumları

göz önüne alındığında UDP protokolünün TCP protokolüne göre daha az güvenli bir iletişim sağladığı sonucu çıkarılabilir.

Aynı anda gönderilen iki tane mektubun aynı gün içerisinde karşı tarafa ulaşma olasılığı vardır. Fakat bu kesinlikle garanti değildir. Aslıda her iki mektubunda karşı tarafa teslim edilmesinin hiçbir şekilde garantisi yoktur. Birisi daha sonraki günde teslim edilecek olabilir. Aynı durum datagram için de geçerlidir. UDP; paketler gönderildiğinde ulaşacağını ya da teslim edilip edilmediğini garanti edemez.

UDP, bağlanma gerektirmeyen ve herhangi bir veri biriminin teslimini veya sıralı oluşunu garanti etmeyen bir teslim hizmeti sunar. Güvenilir iletişime gereksinim duyan bir kaynak ana bilgisayar, TCP hizmeti veya sıraya koyma ve alındı bildirimi hizmetlerini sağlayan bir program kullanmak zorundadır. [26]

UDP ve TCP'nin veri teslim yöntemleri bakımından karşılaştırıldığında; TCP, hedefin erişilebilir ve iletişime hazır olduğunu doğrulayarak bir telefon çağrısı gibi çalışır, UDP ise kart gibi işler; iletiler küçüktür ve teslimat büyük olasılıkla gerçekleşir fakat bu asla garanti edilemez.

UDP genelde bir seferde küçük miktarlarda veri ileten veya gerçek zamanlı işlemlere gereksinim duyan programlar tarafından kullanılır. Bu durumlarda UDP'nin düşük üstbilgi yükü ve çok noktaya yayın yetenekleri (örneğin bir veri birimi için birden çok alıcı) TCP'den daha elverişlidir.

Tablo 4.1'de verilerin taşınmasında UDP veya TCP kullanılmasına bağlı olarak TCP/IP iletişiminin nasıl işlendiği gösterilmektedir.

Tablo 4.1: UDP ve TCP protokollerinin karşılaştırılması

UDP	TCP
Bağlanma olmadan verilen hizmet; ana bilgisayarlar arasında oturum açılmaz.	Bağlanarak verilen hizmet; ana bilgisayarlar arasında oturum açılır.
UDP verilerin sırasını veya teslimini garanti etmez veya doğrulamaz.	TCP, doğrulamayla verinin hedefe ulaşmasını ve verilerin sıralı teslimini garanti altına alır.
UDP kullanan programlar verilerin taşınması için gerekli güvenilirliği kendileri sağlamak zorundadır.	TCP kullanan programlara güvenli veri taşıma garantisi sağlanır.
UDP hızlıdır, ek yükü azdır, noktadan noktaya ve noktan birden çok noktaya iletişimi destekleyebilir.	TCP daha yavaştır, ek yükü fazladır ve yalnızca noktadan noktaya iletişimi destekler.

4.4.1. Soketlerin kullanımı

TCP, telefon görüşmesine benzetildiğinde sokette bir telefona benzetilebilir. Soketler TCP'yi kullanarak iki bilgisayar arasındaki iletişim mekanizmasını sağlar. Bir istemci program iletişiminin sonunda bir soket oluşturur ve bir sunucu ile bağlantı kurmak için girişimde bulunur. Bağlantı kurulduğunda sunucu, bağlantısının sonunda bir soket nesnesi oluşturur. Bundan sonra istemci ve sunucu sokete yazarak ve soketten okuyarak haberleşebilirler.

Soket oluşturmak için bazı kodların yazılması gerekmektedir. Eğer bir istemci bilgisayar programlanmak isteniyorsa öncelikle aşağıdaki şekilde bir soket oluşturulmalıdır:

Socket İstemci;
İstemci = new Socket ("makine adı";PortNumarası);

İstemci bilgisayar değil de sunucu bilgisayar programlanmak isteniyorsa bir soket oluşturmak için gereken kod aşağıda verilmiştir:

Server Socket Servis

```
try {
   MyService = new Server Socket(Portnumarası);}
   catch (IOException e){
   System.out.println(e);}
```

Bir sunucu bilgisayar uygulamaya geçtiğinde, istemciyle bağlantı kurmak için bir soket nesnesi oluşturulmalıdır. Buna ait kodlar aşağıda verilmiştir;

```
Socket istemciSocket=null;
try{
   serviceSocket=MyService.accept();
}
   catch (IOException e){
   System.out.println(e);
}
```

Soket oluşturmak için kodlar yazıldıktan sonra giriş akışını sağlamak için bazı kodlar yazmak gerekmektedir. İstemci bilgisayar üzerinde DataInputStream sınıfını kullanarak sunucu bilgisayardan istemci bilgisayara bilgi giriş akışı sağlanabilinir. Bunun için gerekli kodlar aşağıda verilmiştir:

```
DataInputStream input;
   try {
     input = new DataInputStream(MyClient.getInputStream());
   }
   catch (IOException e) {
     System.out.println(e);
   }
```

DataInputStream sınıfını kullanarak aynı zamanda istemci bilgisayardan giriş alınabilir. Bunun için gerekli kodlar aşağıda verilmiştir:

```
DataInputStream input;
   try {
     input = new DataInputStream(serviceSocket.getInputStream());
```

```
}
catch (IOException e) {
  System.out.println(e); }
```

İstemci bilgisayar üzerinde, PrintStream ya da DataOutputStream sınıflarını kullanarak sunucu sokete bilgi göndermek için çıkış akışı oluşturulabilir. Bunun için PrintStream sınıfıyla oluşturulmuş kodlar aşağıda verilmiştir:

```
PrintStream output;
try {
  output = new PrintStream(MyClient.getOutputStream()); }
catch (IOException e) {
System.out.println(e); }
```

DataOutputStream sınıfıyla oluşturulmuş kodlar aşağıda verilmiştir:

```
DataOutputStream output;
  try {
  output = new DataOutputStream(MyClient.getOutputStream());}
  catch (IOException e) {
  System.out.println(e); }
```

Sunucu bilgisayar üzerinde PrintStream sınıfını kullanarak istemci bilgisayara bilgi gönderilebilir. Bunun için gerekli kodlar aşağıda verilmiştir.

```
PrintStream output;
try {
output = new PrintStream(serviceSocket.getOutputStream()); }
catch (IOException e) {
System.out.println(e);}
```

Soketler kapatılmadan önce mutlaka giriş ve çıkış akışları mutlaka kapatılmalıdır. Sunucu ve istemci bilgisayarlar için gerekle kodlar aşağıda verilmiştir:

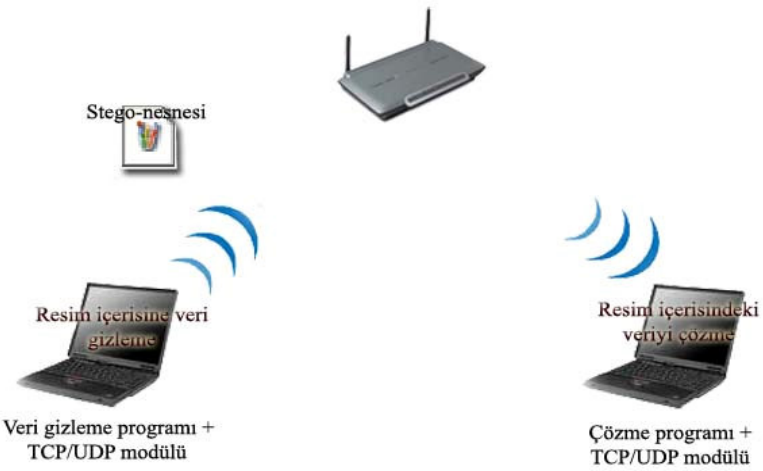
Sunucu bilgisayar için gerekli kodlar:

```
try {
      output.close();
      input.close();
      MyClient.close();
    }
    catch (IOException e) {
      System.out.println(e);
    }
```

Sunucu bilgisayar için gerekli kodlar:

```
    try {
      output.close();
      input.close();
      serviceSocket.close();
      MyService.close();
    }
    catch (IOException e) {
      System.out.println(e);    }
```

4.5. Sayısal Resim İçerisine Veri Gizleme Uygulamaları



Şekil 4.3: Resim içerisine veri gizleme uygulaması

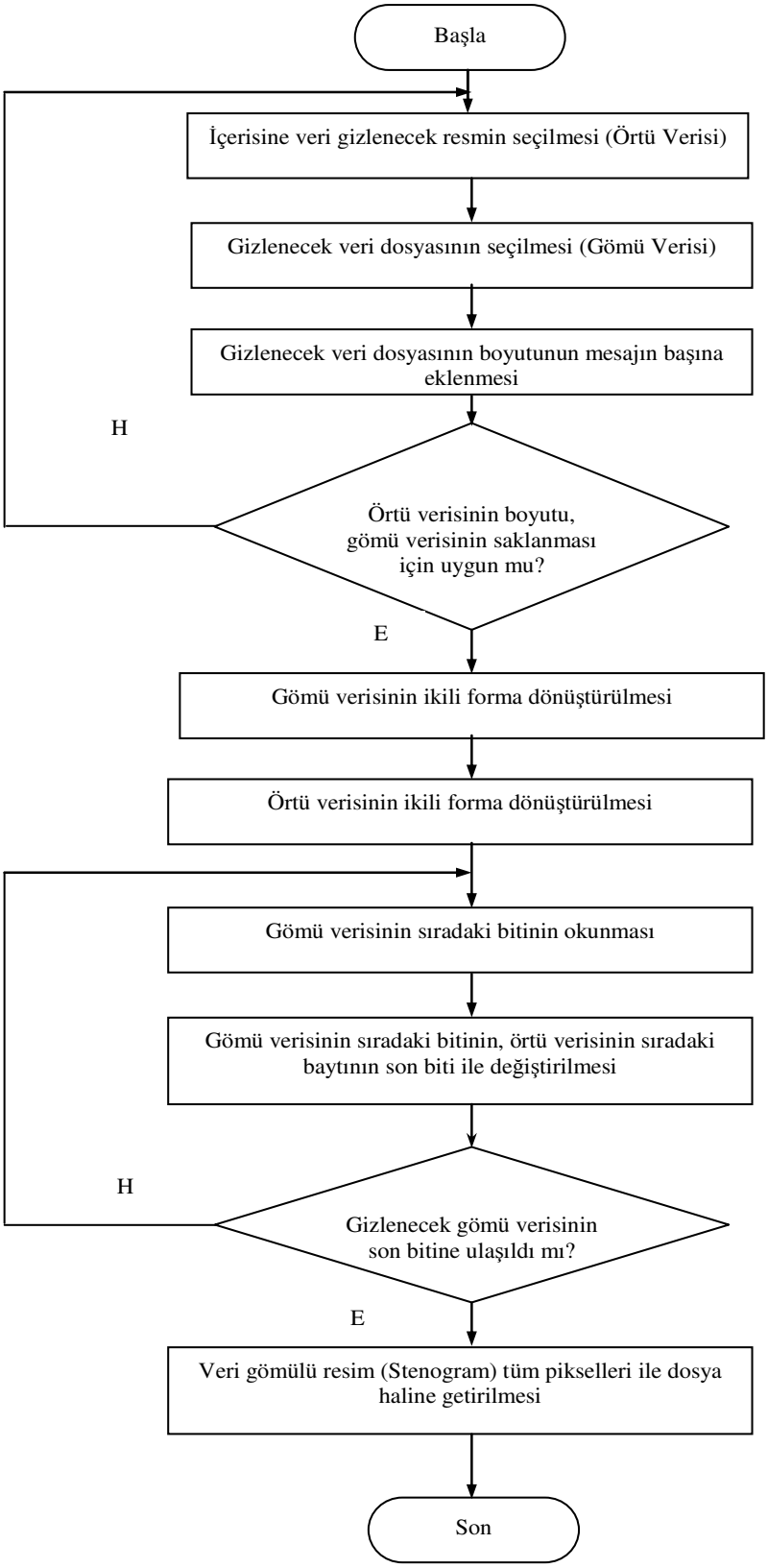
Bu tez çalışmasında, BMP formatında resim içersine gizlenmek istenen veri gömüldükten sonra güvenli bir şekilde kablosuz ağ ortamında bir bilgisayardan başka bir bilgisayara gönderilmesi hedeflenmiştir. Bu hedef doğrultusunda üç farklı yöntem seçeneği ile veri gizleme ve çözme programları hazırlanmıştır. Bu amaca yönelik bir kablosuz ağ ortamı düzeneği kurulmuştur.

Şekil 4.3'de bu düzenek örneklenmiştir. Kaynak bilgisayarda resim içersine veri saklandıktan sonra oluşturulan stego nesnesi hedef bilgisayara kablosuz ortamda güvenli bir şekilde iletilmektedir.

4.6. LSB Yöntemi ile Resim İçerisine Veri Gizleme Uygulamasının Algoritması

Şekil 4.4'deki LSB Yöntemi ile resim içersine veri gizleme uygulamasının algoritması yapısal olarak incelendiğinde uygulamanın çalışması kısaca şöyledir:

- Uygulamanın başlatılması ile birlikte içerisine veri gömülecek olan BMP uzantılı dosya belirlenir.
- Ardından gizli veri dosyası seçilir.
- Gömü verisinin başına boyutu eklenir ve gömü verisi yeniden oluşturulmuş olur.
- Gömen ve gömülen dosyalar arasında kapasite sorunu olup–olmadığı araştırılır. Sonuç olumlu ise resim yani örtü verisi ve gizli veri dosyası yani gömü verisi ikili forma dönüştürülür.
- Gömü verisinin her biti, örtü verisinin sırayla her baytının son bitiyle değiştirilir.
- Bu işlemler gömü dosyanın son bitine ulaşılıncaya kadar sürdürülür.
- Son olarak içerisinde gizli veri gömülü olan resim tüm pikselleri ile dosyaya yazılarak gömme işlemi sonlandırılır.



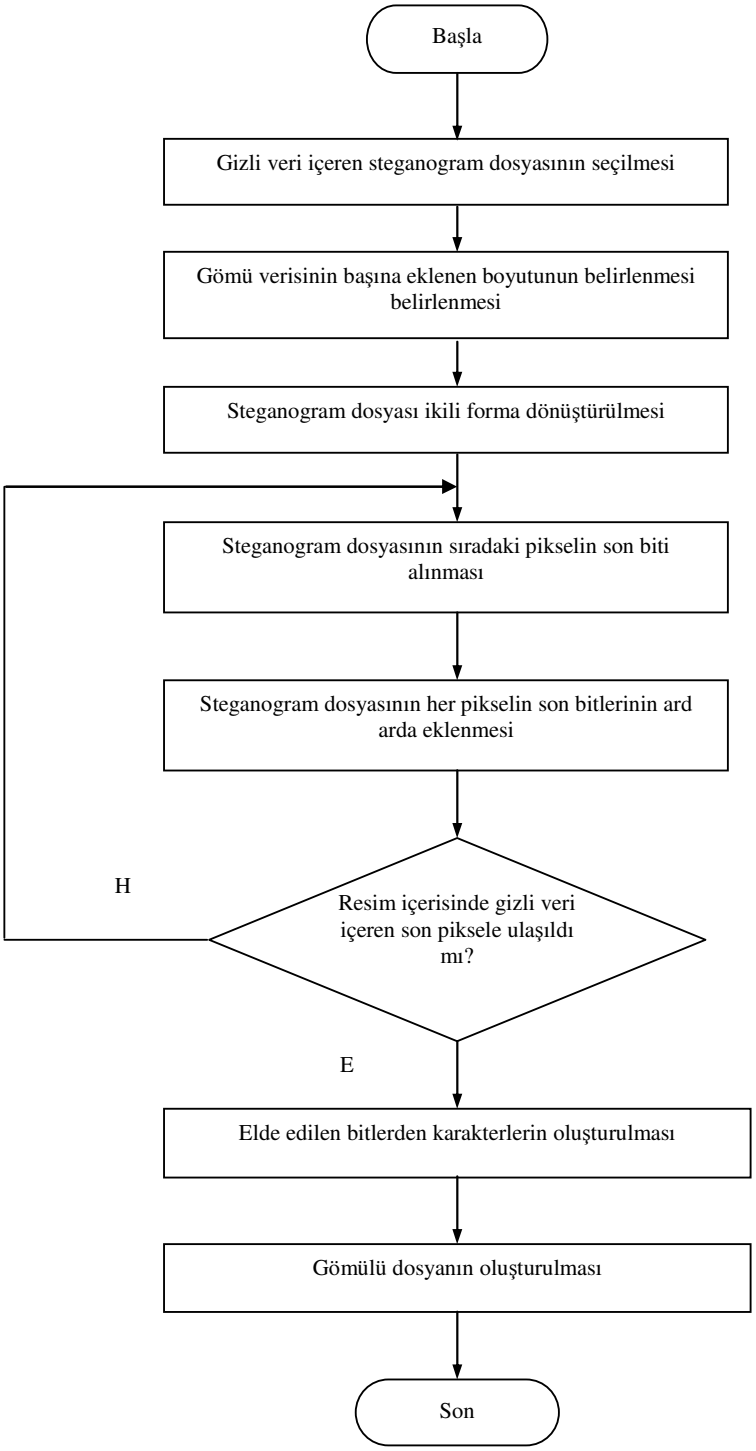
Şekil 4.4: LSB Yöntemi ile resim içersine veri gizleme yönteminin akış diyagramı

4.7. LSB Yöntemi ile İçerisine Veri Gizlenmiş Dosyadan Gömülü Verinin Çıkarılması Uygulamasının Algoritması

Şekil 4.5'deki LSB Yöntemi ile içerisine veri gizlenmiş dosyadan gömü verisinin çıkarılması uygulamasının algoritması yapısal olarak incelendiğinde uygulamanın çalışması kısaca şöyledir:

- Uygulamanın başlatılması ile birlikte içerisinde gizli veri barındıran BMP uzantılı stegonagram dosya belirlenmektedir.
- Daha sonra gömü verisinin boyutu belirlenir. Bu, örtü verisinin ne kadar pikselinden veri elde edileceğini belirler.
- Stegonagram dosyası ikili forma dönüştürülür.
- Stegonagram dosyasının her pikselin son biti alınır. Bu işlem örtü verisinin boyutu kadar devam eder.
- Her 8 bitten bir karakter oluşturulur. Bu şekilde gömülü dosya ortaya çıkar.
- Program sonlandırılır.

.

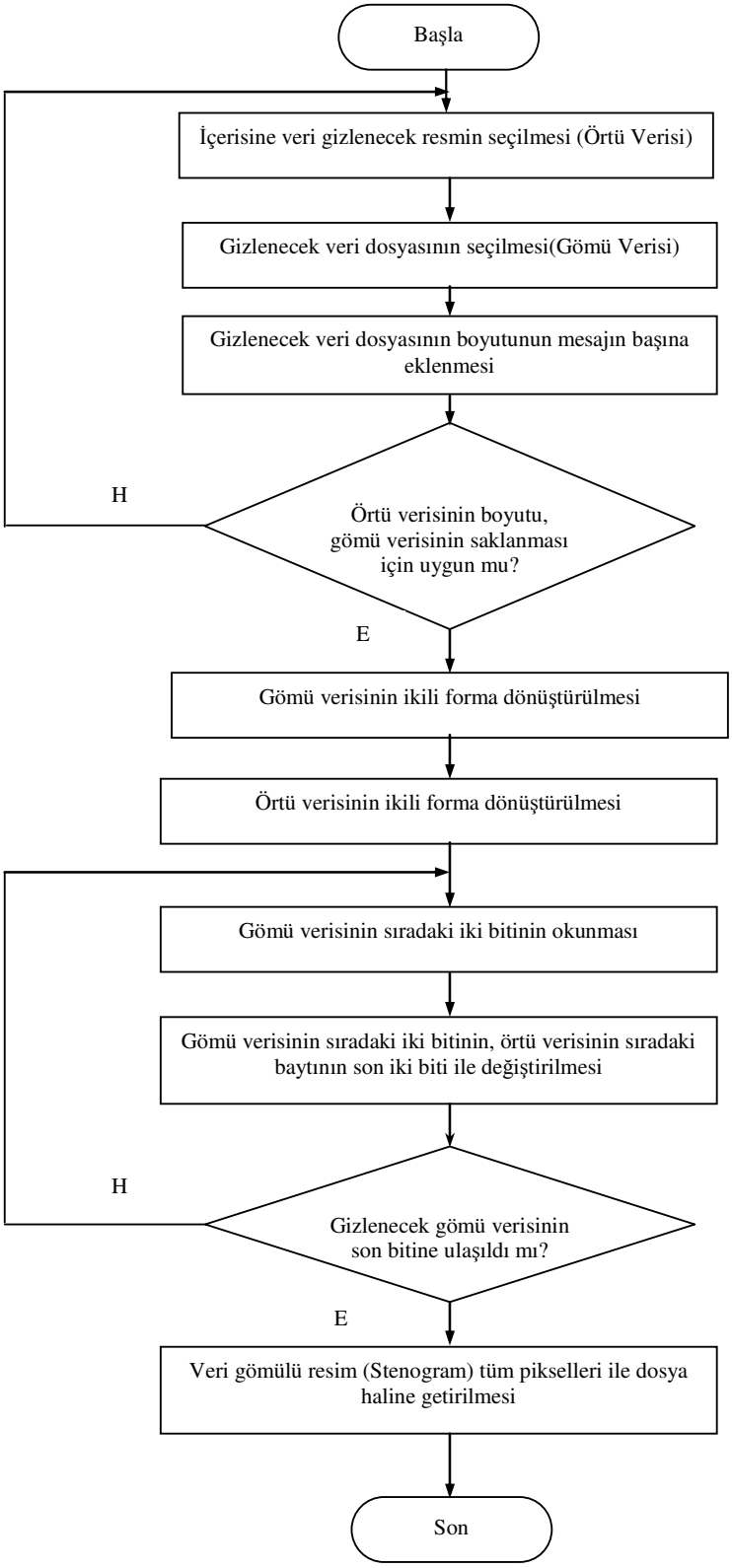


Şekil 4.5: LSB yöntemi ile içerisine veri gizlenmiş dosyadan gömü verisinin çıkarılması uygulamasının akış diyagramı

4.8. İki Bit Yöntemi ile Resim İçerisine Veri Gizleme Uygulamasının Algoritması

Şekil 4.6'daki İki Bit Yöntemi ile resim içersine veri gizleme uygulamasının algoritması yapısal olarak incelendiğinde uygulamanın çalışması kısaca şöyledir:

- Uygulamanın başlatılması ile birlikte içerisine veri gömülecek olan BMP uzantılı dosya belirlenir.
- Ardından gizli veri dosyası seçilir.
- Gömü verisinin başına boyutu eklenir ve gömü verisi yeniden oluşturulmuş olur.
- Gömen ve gömülen dosyalar arasında kapasite sorunu olup–olmadığı araştırılır. Sonuç olumlu ise resim yani örtü verisi ve gizli veri dosyası yani gömü verisi ikili forma dönüştürülür.
- Gömü verisinin sırayla her iki biti, örtü verisinin sırayla her baytının son iki bitiyle değiştirilir.
- Bu işlemler gömü dosyanın son bitine ulaşılıncaya kadar sürdürülür.
- Son olarak içerisinde gizli veri gömülü olan resim tüm pikselleri ile dosyaya yazılarak gömme işlemi sonlandırılır.

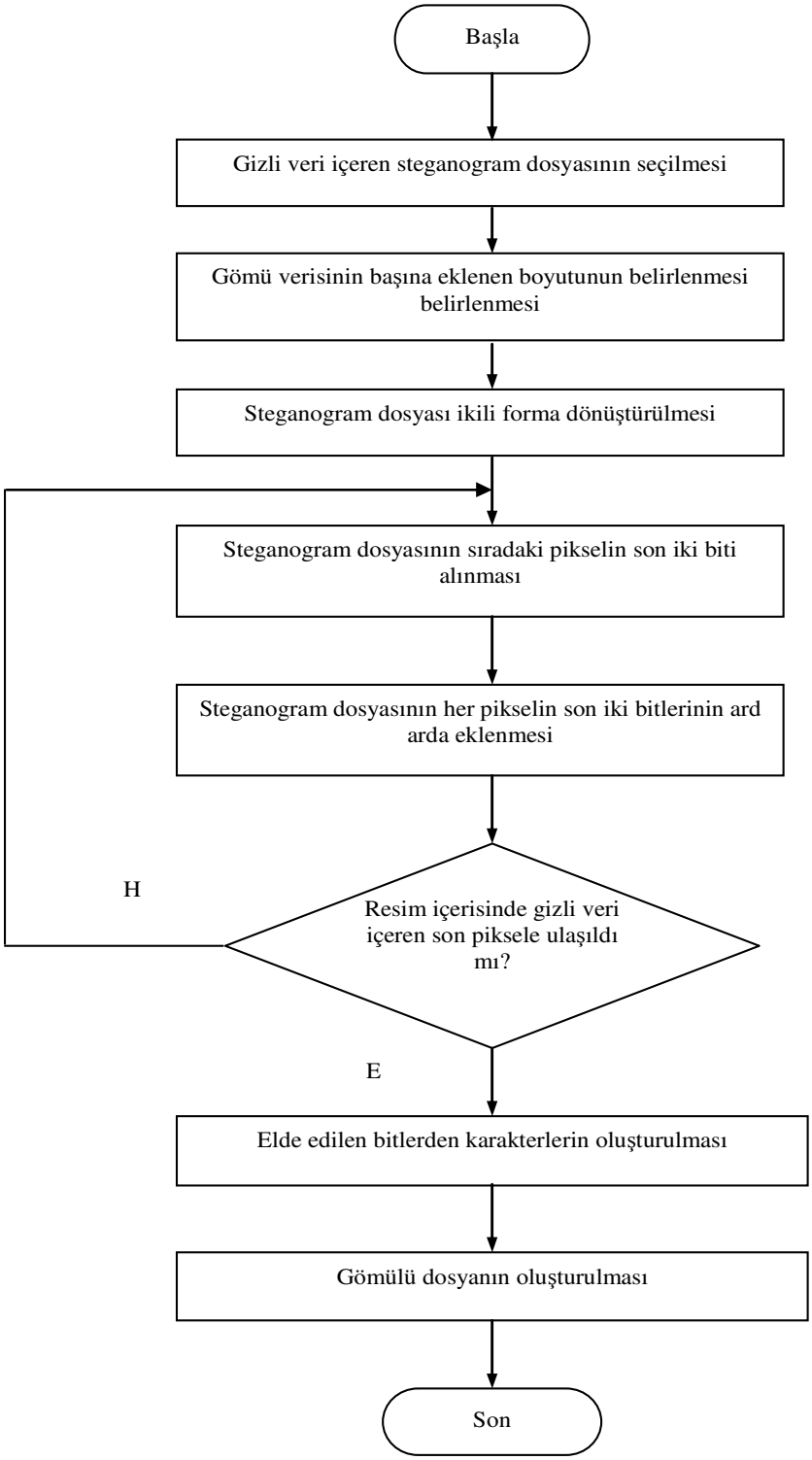


Şekil 4.6: İki Bit Yöntemi ile resim içersine veri gizleme yönteminin akış diyagramı

4.9. İki Bit Yöntemi ile İçerisine Veri Gizlenmiş Dosyadan Gömü Verisinin Çıkarılması Uygulamasının Algoritması

Şekil 4.7'deki İki Bit Yöntemi ile içerisine veri gizlenmiş dosyadan gömü verisinin çıkarılması uygulamasının algoritması yapısal olarak incelendiğinde uygulamanın çalışması kısaca şöyledir:

- Uygulamanın başlatılması ile birlikte içerisinde gizli veri barındıran BMP uzantılı stegonagram dosya belirlenmektedir.
- Ardından gömü verisinin boyutu belirlenir. Bu, örtü verisinin ne kadar pikselinden veri elde edileceğini belirler.
- Stegonagram dosyası ikili forma dönüştürülür.
- Stegonagram dosyasının her pikselin son iki biti alınır. Bu işlem örtü verisinin boyutu kadar devam eder.
- Her 8 bitten bir karakter oluşturulur. Bu şekilde gömülü dosya ortaya çıkar.
- Program sonlandırılır.

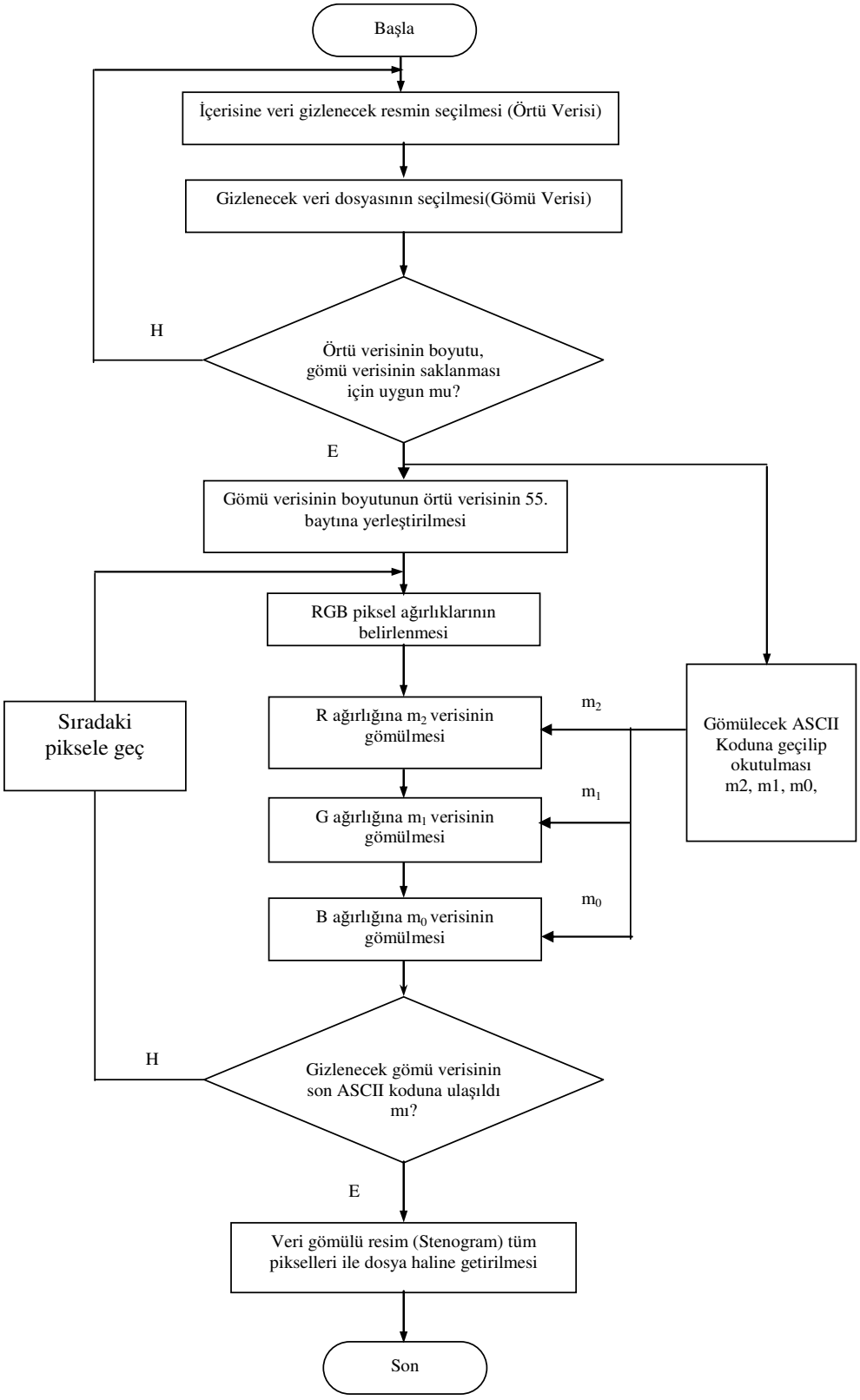


Şekil 4.7: İki Bit Yöntemi ile içerisine veri gizlenmiş dosyadan gömü verisinin çıkarılması uygulamasının akış diyagramı

4.10. Bir Piksel İçerisine bir ASCII Kodun Gömülmesi Yöntemi ile Resim İçerisine Veri Gizleme Uygulamasının Algoritması

Şekil 4.8'deki Bir Piksel İçerisine bir ASCII Kodun Gömülmesi Yöntemi ile resim içersine veri gizleme uygulamasının algoritması yapısal olarak incelendiğinde uygulamanın çalışması kısaca şöyledir:

- Uygulamanın başlatılması ile birlikte içerisine veri gömülecek olan BMP uzantılı dosya belirlenir.
- Ardından gizli veri dosyası seçilir.
- Gömen ve gömülen dosyalar arasında kapasite sorunu olup–olmadığı karşılaştırılır.
- Gömü verisinin boyutu örtü verisinin 55. baytına yerleştirilir.
- Resmin RGB (m_0, m_1, m_2) piksel ağırlıkları belirlenir. Aynı anda gömülecek dosyanın ASCII kodları da belirlenmektedir.
- Resmin RGB ağırlıklarının son rakamlarına sırasıyla gizlenen verinin ASCII değerleri atanır.
- Bu işlemler gizli dosyanın son ASCII koduna ulaşılıncaya kadar sürdürülür.
- İçerisinde gizli veri gömülü olan resim tüm pikselleri ile dosyaya yazılarak kaydedilir.

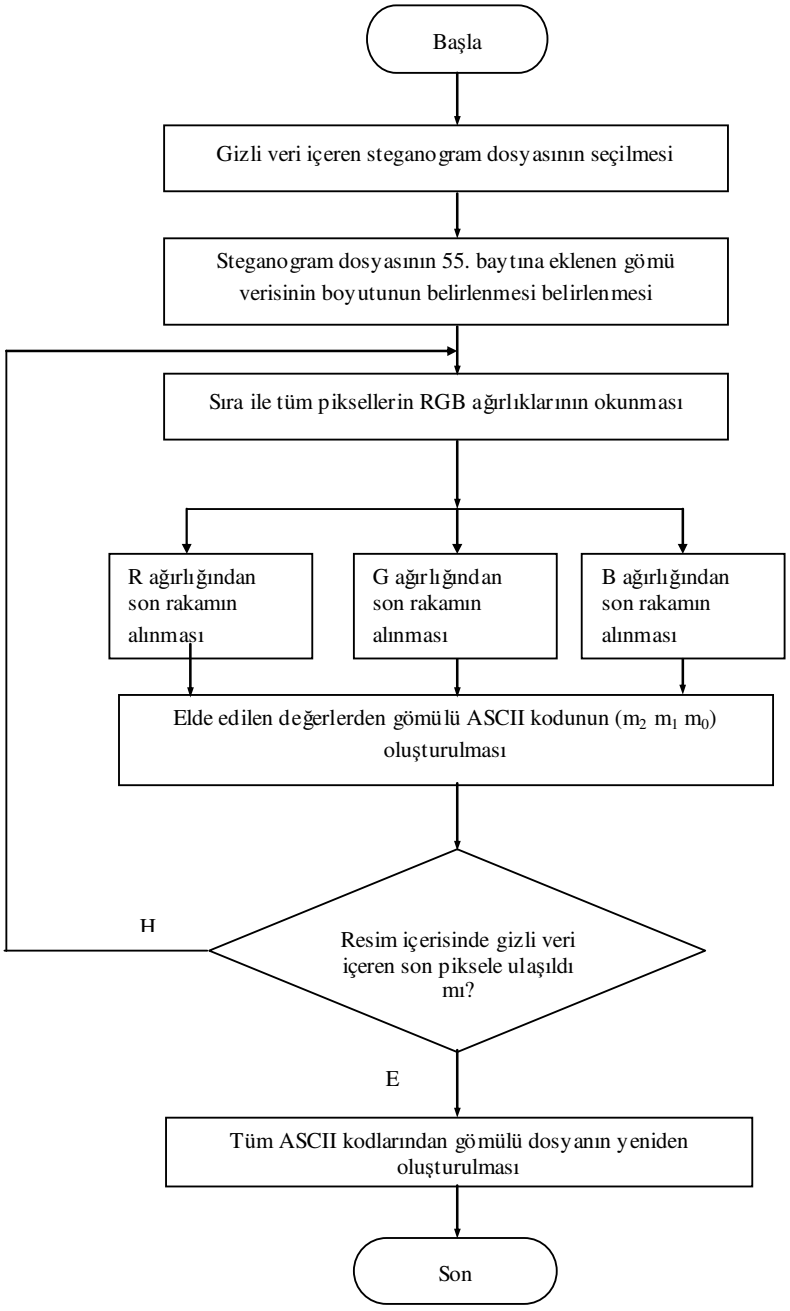


Şekil 4.8: Bir Piksel İçerisine bir ASCII Kodun Gömülmesi Yöntemi ile resim içerisine veri gizleme uygulamasının akış diyagramı

4.11. Bir Piksel İçerisine bir ASCII Kodun Gömülmesi Yöntemi ile Dosyadan Gömü Verisinin Çıkarılması Uygulamasının Algoritması

Şekil 4.9'daki Bir Piksel İçerisine bir ASCII Kodun Gömülmesi Yöntemi ile içerisine veri gizlenmiş dosyadan gömü verisinin çıkarılması uygulamasının algoritması yapısal olarak incelendiğinde uygulamanın çalışması kısaca şöyledir:

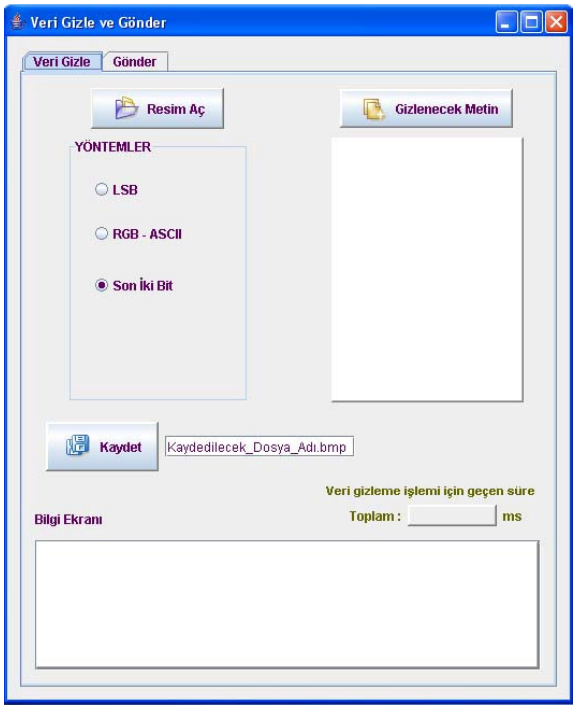
- Uygulamanın başlatılması ile birlikte içerisine veri gömülecek olan BMP uzantılı dosya belirlenir.
- Steganogram dosyasının 55. baytına yerleştirilen gömü verisinin boyutu belirlenir.
- Ardından RGB piksel ağırlıkları okunarak her bir ağırlığın son rakamında gömülü olan veriler alınarak (m_0,m_1,m_2) ASCII kodu oluşturulur.
- Elde edilen kod dosyaya yazılarak işlemler gizlenen dosyanın sonuna kadar sürdürülür.
- Tüm ASCII kodlarından gizli veri dosyası oluşturularak disk üzerine kaydedilerek, program sonlandırılır.



Şekil 4.9: Bir Piksel İçerisine bir ASCII Kodun Gömülmesi Yöntemi ile dosyadan gömü verisinin çıkarılması uygulamasının akış diyagramı

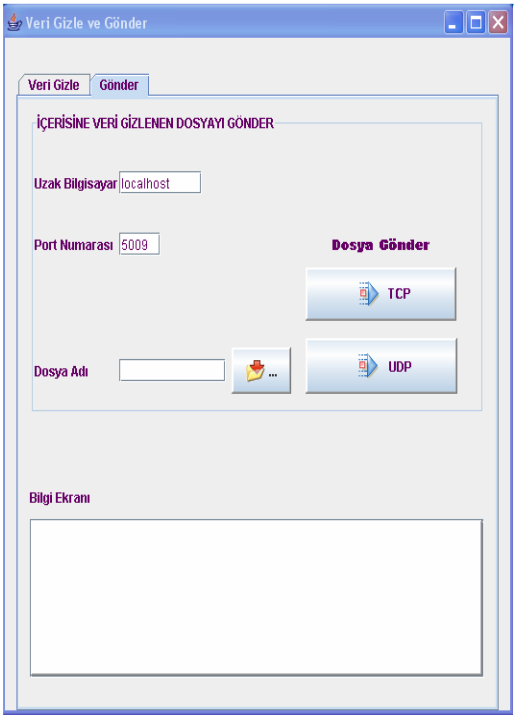
4.12. Resim İçerisine Metin Gizleme/Çözme Örnek Uygulamaları

Bu çalışmada gizleme objesi olarak BMP formatında resim, gömü dosyası olarak da metin dosyası kullanılmıştır. Bu uygulama çalışmasında veri gizleme için bir ve gizlenen verinin yeniden elde edilmesi için bir toplam iki program bulunmaktadır. Veri gizleme işlemi bir bilgisayarda hazırlanıp uzak bilgisayara gönderilmektedir. Gönderme işlemi başlamadan önce karşı bilgisayarın veriyi almak üzere hazır bekleme durumuna geçmesi gerekmektedir. Programın her penceresinde alt kısımda, yapılan her adımı sıralamak üzere bilgi ekranları bulunmaktadır.



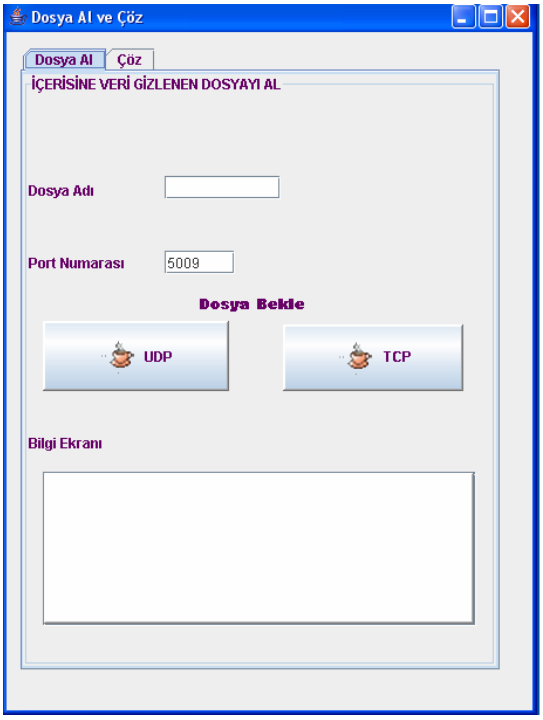
Şekil 4.10: Resim içerisine metin gizleme arabirimi

Programın ilk bölümü "Veri Gizle ve Gönder" olmak üzere iki kısımdan oluşmaktadır. Şekil 4.10'da görüldüğü gibi ilk kısımda içersine veri saklanacak resim ve gizlenmek istenilen veri belirlenmektedir. Veri saklamak için üç tane farklı yöntem geliştirilmiştir. Veri saklama işlemine geçilmeden önce burada verinin hangi yolla saklanacağı seçilmelidir.



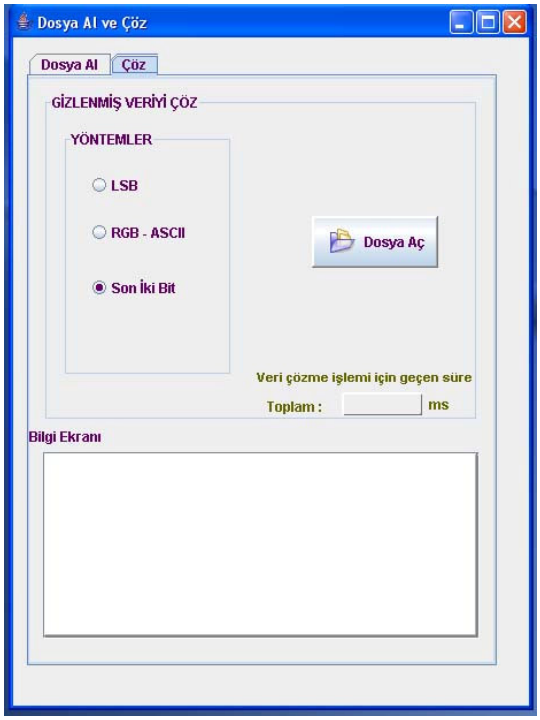
Şekil 4.11: Resim gönderme arabirimi

Şekil 4.11'de programın veri gizleme ve gönderme bölümünün ikinci kısmı görülmektedir. Bu kısımda gönderilecek resim belirlendikten sonra resmin gönderileceği bilgisayar, port numarası ve hangi protokol ile gönderileceği belirlenmelidir.



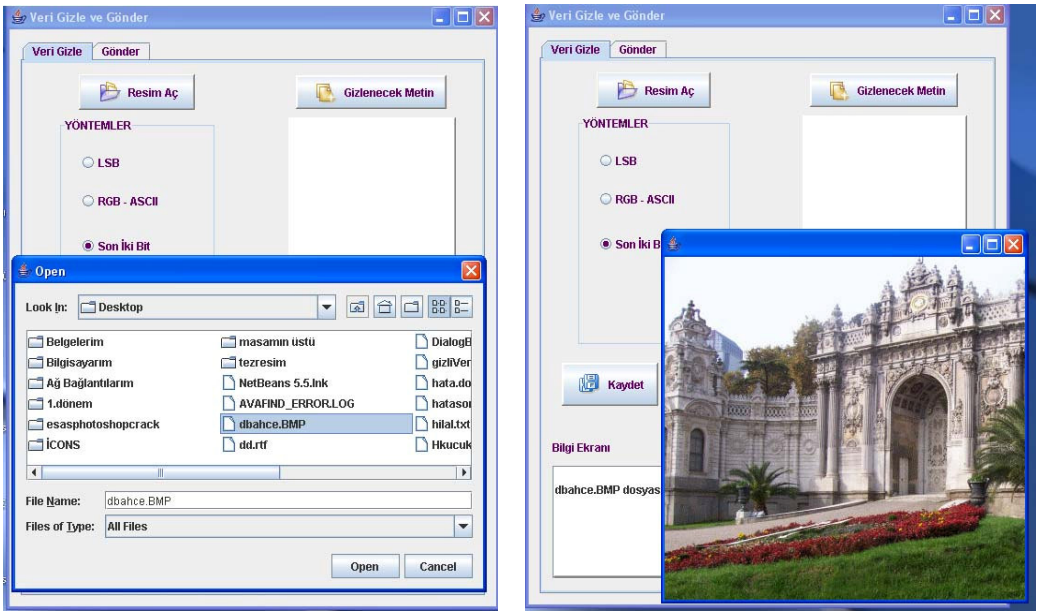
Şekil 4.12: Resim alma arabirimi

Programın ikinci bölümü ise "Veri Al ve Çöz" olmak üzere iki kısımdan oluşmaktadır. Karşı tarafta resmin içersine veri saklama işlemi gerçekleştikten sonra resmi alacak bilgisayarın bekleme konumuna geçmesi gerekmektedir. Şekil 4.12'de görüldüğü gibi önce alınacak resim için bir isim ve port numarası belirlenmelidir. Daha sonra dosya almak için uygun protokol seçilip ve "Dosya Bekle" butonuna basılarak karşı tarafın resmi göndermesi beklenmelidir.



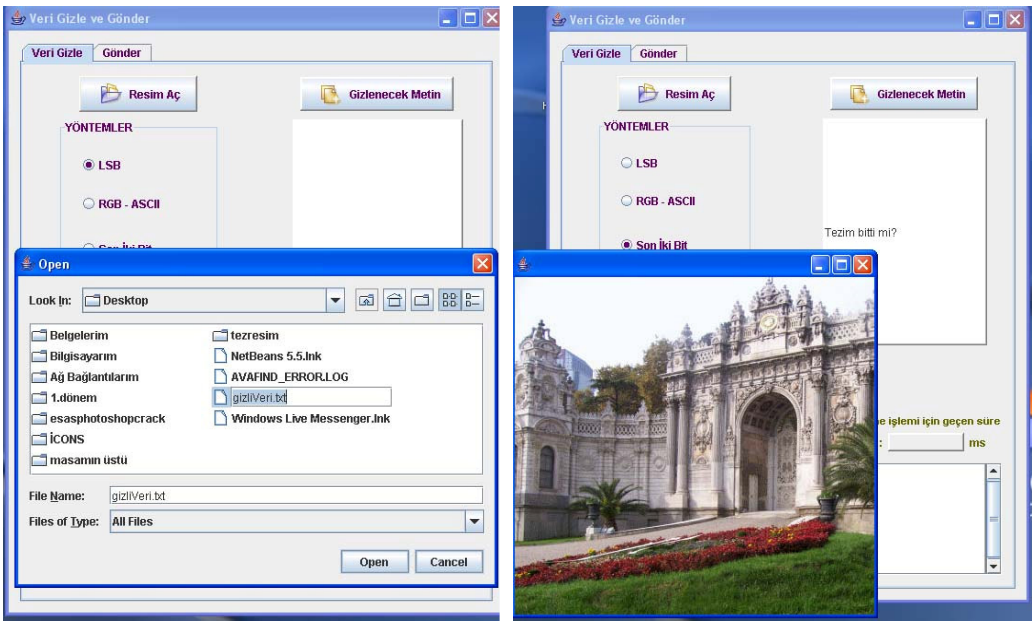
Şekil 4.13: Resim içerisinden metini çözme arabirimi

Şekil 4.13'de programın "Veri Al ve Çöz" bölümünün ikinci kısmı görülmektedir. Veri saklamak için üç tane farklı yöntem geliştirilmiştir. Veri çözme işlemine geçilmeden önce burada verinin hangi yolla çözüleceği seçilmelidir. Bekleme süreci bitip dosya hedefe ulaştıktan sonra "Doya Aç" butonuna basılır. Buradan bir önceki bölümde kaydedilen dosyayı açarak gizli veriye ulaşılmaktadır.



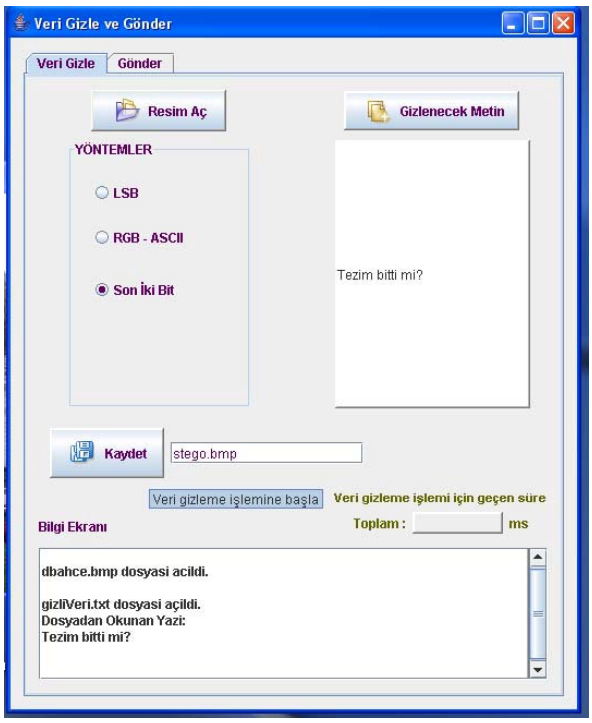
Şekil 4.14: Veri gömülecek resmin seçilmesi ve veri gömülecek resim

Yapılan örnek uygulamada “Resim Aç” butonuna basıldıktan sonra örtü verisi olarak dbahce.bmp resim doyası belirlenmiştir. Şekil 4.14’de “Resim Aç” butonuna basıldıktan sonraki durum gösterilmektedir. Örtü veri olarak seçilen dbahce.bmp resim dosyası seçilmiştir.



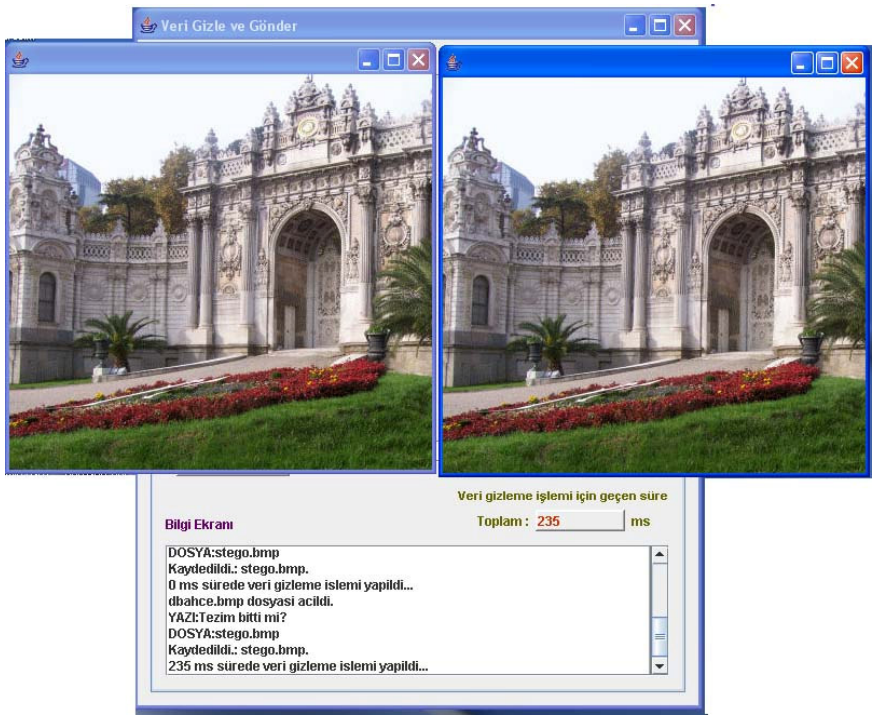
Şekil 4.15: Resmin içersine gömülecek verinin seçilmesi ve gömülecek veri

Örtü verisi seçildikten sonra gizlenecek gömü verisi seçilmelidir. Bunun için "Gizlenecek Metin" butonuna basılmalıdır. Şekil 4.15'de "Gizlenecek Metin" butonuna basıldıktan sonraki durum ve gömü verisi olarak seçilen gizliVeri.txt metin doyası görülmektedir. Bu metin dosyasının içeriği aynı zamanda bizim resim içerisine saklamak istediğimiz metindir. Bu uygulama için "Tezim bitti mi?" gibi örnek bir metin seçilmiştir.



Şekil 4.16: Gizli metin içeren resmin kaydedilmesi

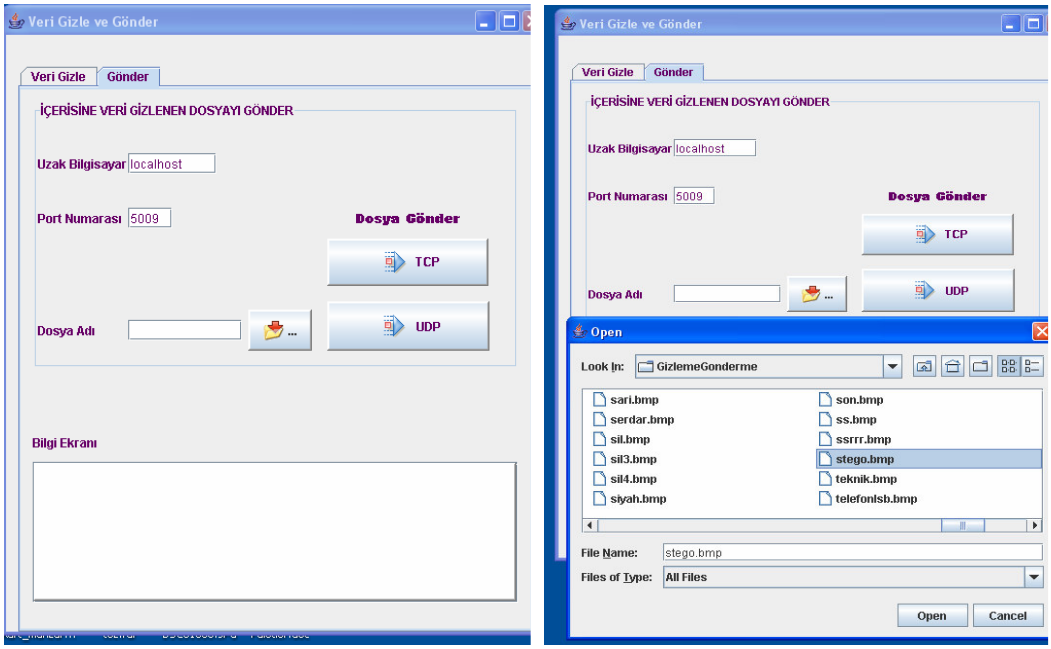
Veri gizleme işlemini gerçekleştirmek için gerekli olan resim, metin ve kullanılacak yöntem seçildikten sonra veri saklama işlemini gerçekleştirmek için Şekil 4.16'da görüldüğü gibi içerisine veri gizlenmiş resme yani stego nesnesine bir isim verildikten sonra kaydet butonuna basılmalıdır. Bu uygulama için stego nesnesine stego.bmp adı verilmiştir.



Şekil 4.17: Orijinal resim ile gömme İşlemi sonucunda oluşan resmin karşılaştırılması.

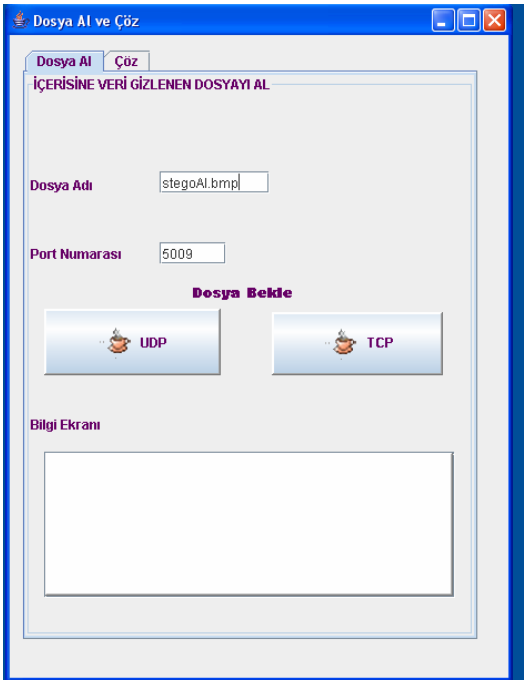
"Kaydet" butonuna basıldıktan sonra veri saklama süreci başlar. Verilen yeni isimle stego nesnesi oluşturulur ve bu zaman zarfında geçen süre hesaplanır. Şekli 4.17'de "Kaydet" butonuna basıldıktan sonra karşımıza çıkan resmin ilk ve veri saklandıktan sonraki durumları görülmektedir. Bu işlem Şekil 4.17'de görüldüğü gibi 235 ms' de tamamlanmıştır. İki resim karşılaştırıldığında aradaki farkı gözün algılayamayacağı ortadadır.

Veri gizleme işlemi tamamlanıp stego nesnesi oluşturulduktan sonra veri gönderme işlemine geçilebilir. Programın Veri Gizleme ve Gönderme Bölümünün ikinci kısmına geçmek gerekmektedir.



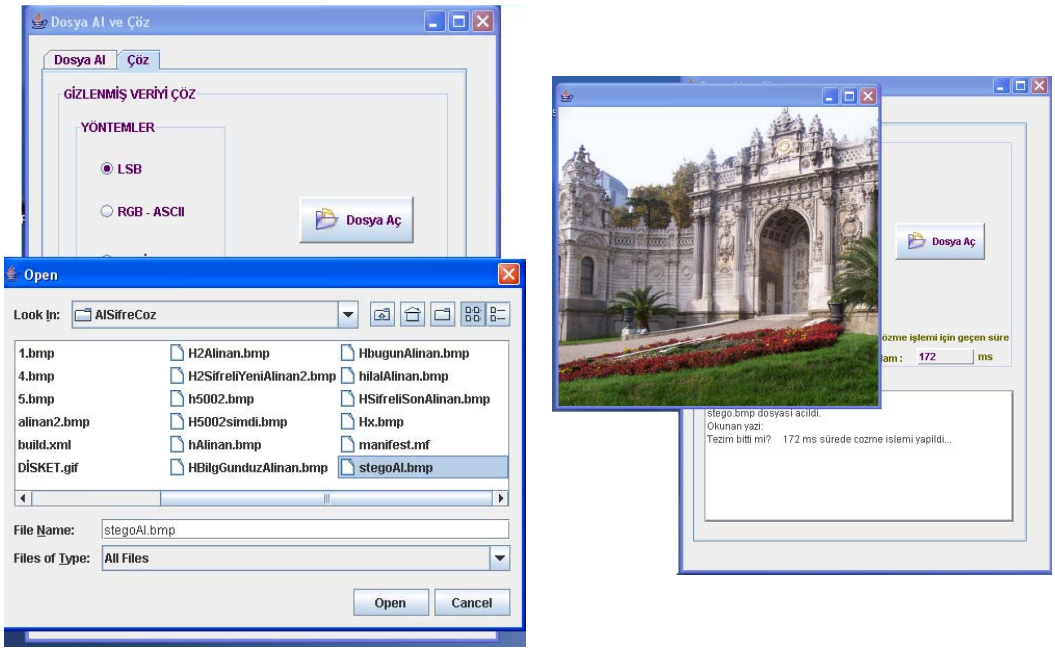
Şekil 4.18: İçersine veri gömülen resmi gönderme arabirimi ve resmin seçilmesi

Uygulama programının Veri Gizleme ve Gönderme Bölümünün ikinci kısmı olan "Gönder" bölümü Şekil 4.18'de görülmektedir. Stego nesnesini göndermek için ilgili butona basıldıktan sonra Şekil 4.18'de görüldüğü gibi ilgili resim dosyası seçilmelidir. Bizim örnek uygulamamız için bu dosya bir önceki adımda kaydettiğimiz stego.bmp dosyasıdır. Bu bölümde belirlenen dosyayı göndermeden önce dosyayı alacak tarafın bekleme durumuna geçmesi gerekmektedir.



Şekil 4.19: İçersine veri gizlenmiş resmi bekleme

İçerisine veri saklandıktan sonra stego nesnesini alacak olan bilgisayar, resmi almak için Şekil 4.19'da görüldüğü gibi gelen dosyaya bir isim verdikten sonra dosyanın gönderildiği protokole göre "Dosya Bekle" butonlarından birine basmalıdır. Bu uygulama örneği için beklenen dosyaya stegoAl.bmp adı verilmiştir. Bu işlemden sonra gönderici bilgisayar Şekil 4.18'de görülen "Dosya Gönder" butonuna bastığında dosya gönderme işlemi tamamlanmış olur.



Şekil 4.20: Gizli veri içeren resmin açılması ve gizli veri

Stego nesnesi belirlenen isimle alındıktan sonra içerisine gizlenmiş veriyi çözme işlemine gerçekleştirilebilir. Bunun için programın "Çöz" bölümüne geçilmesi gerekmektedir. Veriyi çözme yöntemi belirlendikten sonra Şekil 4.20'de görüldüğü gibi "Dosya Aç" butonuna basılarak bir önceki adımda belirlenin ve stegoAl.bmp olarak adlandırılan stego nesnesi açılır. Açılacak dosya belirlendikten sonra veri çözme işlemi başlar ve bu işlemin gerçekleşme süresi hesaplanır.

Şekil 4.20'de Stego nesnesi, saklı veri ve geçen süre görülmektedir. Çözme işlemi için 172 ms geçmiştir. Gömü verisi olan "Tezim bitti mi?" metni çözülmüştür.

4.13. Veri Gizleme Yöntemlerinin Karşılaştırmalı Başarım Değerlendirmesi

Analiz süresince 2 farklı bilgisayar kullanılmış olup, bunlara ait donanım özellikleri aşağıda verilmiştir.

1. Bilgisayar: Celeron 1.72 Ghz işlemci, 512 Mb RAM

2. Bilgisayar: Core 2 Duo 1.83 Ghz işlemci, 1Gb RAM

8 Kbyte gizli veri dosyasının, 900Kbyte büyüklüğündeki bmp uzantılı bir resim dosyası içerisine gömülmesi işlemi tüm uygulamalar için gerçekleştirilerek elde edilen başarım sonuçları Tablo 4.2'de gösterilmiştir.

Tablo 4.2: Uygulamaların zaman performans ölçümü

Yöntemler	Gömülen dosyanın boyutu	Gömme işlem süresi (ms)		Çözme işlem süresi (ms)	
		1.Bilgisayar	2. Bilgisayar	1.Bilgisayar	2. Bilgisayar
LSB	8 KB	512	235	750	204
RGB– ASCII	8 KB	359	125	312	195
İKİ BİT	8 KB	360	156	297	78

Tablo 4.2'deki değerler incelendiğinde; bilgisayar donanım özellikleri (işlemci türü, hızı, RAM bellek) iyileştikçe şifreleme ve şifre çözme zamanının azaldığı görülmektedir.

5. SONUÇ VE ÖNERİLER

Günümüzde güvenlik gibi güncel gereksinimler açısından veri gizleme üzerine yapılan çalışmalar, büyük önem kazanmaktadır. Verinin gizli, güvenli ve hızlı bir şekilde hedefe ulaştırılması ihtiyacı günümüzde birçok alanda karşımıza çıkmaktadır.

Bu çalışmanın amacı; LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemlerinden ve UDP/TCP protokollerinden yararlanarak kablosuz ortamda, bilginin korunması ve güvenle istenen hedefe ulaştırılmasının sağlanmasıdır. Kablosuz ortamın en büyük problemlerinden ikisi güvenlik ve sınırlı bant genişliğidir. Gerçekleştirilen yöntemler ile resim üzerinde gözle görülebilir bir bozulma oluşmadan gizli bilgiyi göndermek ve bant genişliğini daha etkin bir şekilde kullanımı sağlanmıştır.

Araştırmada kullanılan LSB gömme yöntemi ile resmi oluşturan tüm piksellerin son bitlerini, gizlenmek istenen mesajın bitleriyle yer değiştirerek gizli mesajın resim içerisine saklanması sağlanmıştır. Böylelikle değiştirilen bit en az öneme sahip olan bit olduğu için ilgili pikselin renginde gözün algılayabileceği bir değişiklik ortaya çıkmamıştır. İki bit yönteminde ise gizli verinin bitleri sırasıyla ikişer ikişer resmin her baytının son iki bitiyle yer değiştirilmiştir. Bu sayede LSB'ye göre daha fazla veri resim içerisine saklanmıştır. RGB – ASCII yönteminde ise bit düzeyinde değil ASCII değerlerle çalışılmıştır. Saklanmak istenen verinin ASCII karşılıkları her baytının ASCII değerlerine belli bir algoritmayla gömülmüştür.

Bu üç yöntemde de resim içersine fazladan bir bilgi eklenmeyip, sadece ilk iki yöntemde bazı bitlerinde, üçüncü yöntemde ise ASCII değerlerinde değişiklik yapıldığı için resmin dosya boyutunda da bir artış söz konusu olmamaktadır. Bu önemli bir ayrıntıdır. Çünkü elde edilen resmin sınırlı bant genişliğine sahip kablosuz ortamdan iletilmesi hedeflenmektedir. Böyle bir uygulama ile aslında, aynı anda iki dosya birden gönderilmektedir: içerisine veri gizlenen resim dosyası ve gizlemek

istenen bilgi. Böylelikle sınırlı bant genişliğinin daha etkin kullanılması da sağlanmaktadır.

Araştırma kapsamında veri gizleme tekniklerinden LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi ile veri gizlenmiş resmin, kablosuz ortamdan TCP ve UDP protokolleriyle gönderilmesi Java yazılım dili ile örnek bir uygulama gerçekleştirilerek, elde edilen sonuçlar ve bu yöntemlerin gerçekleme açısından zorluk dereceleri karşılaştırılmıştır.

Tez çalışmasında yapılanlar ve elde edilen sonuçlar özetle şunlardır:

- LSB, İki Bit ve Bir Piksel İçerisine bir ASCII Kodun Gömülmesi yöntemleri kullanarak BMP formatındaki bir resim içerisine herhangi bir metin dosyasındaki veriyi gizleyerek yeni bir resim dosyası oluşturulmuştur.
- Gizlenmiş bilgiyi içeren resim dosyasını kablosuz ortamdan TCP ve UDP protokollerini kullanarak transfer edilmiştir.
- Gizlenmiş bilgiyi içeren resim dosyasından iletilmesini istediğimiz gizli bilgi tekrar çıkartılmıştır.
- Gönderilmek üzere oluşturulan ve bilgimizi de içeren yeni resim dosyasında gözün algılayabileceği herhangi bir değişiklik olmamaktadır.
- Veri gizlendikten sonra resmin boyutunda hiçbir değişiklik oluşmamıştır.
- Resim, ağ ortamında iletileceği için gönderilen mesajın boyutu, hızlı iletim bakımından önem taşımaktadır. Çalışmalarda sabit bir resim içerisine hangi yöntemle daha çok bilgi saklanabileceği tespit edilmiştir.
- Bu uygulamanın önemli bir avantajı, aynı anda iki veri dosyası gönderildiğinden ve boyut sadece resim dosyasının boyutu olduğundan bant genişliğinin daha etkin kullanılmasıdır.
- Bu uygulamanın bir dezavantajı ise kapasite sınırının olmasıdır. Resmin boyutunun mutlaka gizlenmek istenen verinin boyutunu karşılıyor olması gerekmektedir. Ayrıca bu uygulamada BMP formatında resim kullanılmaktadır. BMP formatındaki resimlerin ilk 54 bayt'ı başlık kısmıdır. Bu nedenle veri gizleme işlemine 55. bayt'dan başlanmıştır.

- Veri saklama ve çıkarmada; üç değişik yöntemin uygulamalarına ait Tablo 4.2'de gösterilen sonuçlara bakıldığında, süre bakımından İki Bit yöntemi en hızlı yöntem olduğu ortaya çıkmaktadır.
- Saklanan gizli verinin boyutu göz önüne alındığında en avantajlı yöntem RGB–ASCII yöntemidir. RGB–ASCII yönteminde 3 bayt veriye 1 bayt, İki Bit yönteminde 8 bayt veriye 2 bayt, LSB yönteminde ise 8 bayt veriye; 1 bayt veri saklanabilmektedir.

Öneriler:

- Örtü verisi olarak ses ya da video kullanılabilir.
- Gömü verisi olarak resim ya da ses kullanılabilir.
- Veri gizlemek için sıkıştırılmış resim formatları (jpeg ,gif, png) kullanılabilir.
- Veriler sıkıştırılarak saklanabilir.

KAYNAKLAR

[1] Kutucu, H., Kaya, M., Dalkınç, M., Criptography and Network Security, ***Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü*** 1–2, (2002).

[2] Katzenbeisser, S., Petitcolas, A.P., Information Hiding Techniques for Steganografik and DigitalDamgalama, 1, ***New york.Artech House, Computer Security Series,*** (2002).

[3] Sözlük, 2006, Hermes, http://www.dijital–imza.com/sozluk.htm **(Ziyaret tarihi:30 Ekim 2006).**

[4] Turk İnternet Com haber Merkezi, 2003, Tursing A.S'den "Digital Damgalama", The Internet & IT Network, http://turk.internet.com/haber/yazigoster.php3?yaziid= 7847 **(Ziyaret tarihi:15 Ekim 2006).**

[5] Johnson N.F., Jajodia S., ***Steganalysis: The Investigation of Hidden Information, IEEE Informatıon Technology Conference***, pp. 113–116, New York, USA, (1998).

[6] Johnson N.F., Jajodia S., ***Exploring Steganografik: Seeing the Unseen, IEEE Computer***, pp. 26–34, New York, USA, (1998).

[7] F.Akar, H.Selçuk Varol, ***A New RGB Weighted Encoding Technique for Efficient Information Hiding in Images, Journal of Naval Science and Engineering***, Volume 2, 21–36, (2004).

[8] Trithemius, J., Steganographia, 3, Jim Reeds, ***AT&T Labs , Research Florham Park, New Jersey,*** (1464–1516).

[9] Schotti, G., Steganographica, (1665), ***History of Steganografik and Cryptography***, http://www.petitcolas.net/fabien/steganografik/history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[10] Hayhurst, J.D., ***The Pigeon Post into Paris*** (1870–1871), History of Steganografik and Cryptography, http://www.petitcolas.net/fabien/ steganografik /history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[11] Kerckhoffs, A., ***Cryptographie Militaire***, (1883), History of Steganografik and Cryptography, http://www.petitcolas.net/fabien/steganografik/history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[12] Briquet, C.M., Les Filigranes, ***Geneva*** , (1907), History of Steganografik and Cryptography, http://www.petitcolas.net/fabien/steganografik/history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[13] Leary, T., ***Cryptology in the 15th and 16th century***, (1996), History of Steganografik and Cryptography , http://www.petitcolas.net/fabien/steganografik/ history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[14] Kelly, T., ***The Myth Of The Skytale***, , Cryptologia V XXII No 3, Pp 44–260, (1998), History of Steganografik and Cryptography, http://www.petitcolas.net/ fabien/steganografik/history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[15] Reeds J., Cryptologia, , Solved: The Ciphers in Book 3 Of Trithemius'steganographiav XXII No 4, pp 291–318, (1998), History of Steganografik and Cryptography, http://www.petitcolas.net/fabien /steganografik /history.html, **(Ziyaret tarihi:21 Kasım 2006).**

[16] Şahin, A., Buluş, E., Sakallı, M.T.,Gri Seviye Resimler Üzerine Rastgele LSB Yöntemini Ve Sayı Teorisini Kullanarak Bilgi Gizleme Ve Steganaliz, ***Trakya Üniversitesi Fen Bilimleri Enstitüsü***,1–3, (2006).

[17] Sarıoğlu, Ş., Tunçkanat, M., 2002, Güvenli Internet Haberleşmesi İçin Bir Yazılım: TurkSteg, Olympos Security, http://www.teknoturk.org/ docking/yazilar/ tt000106–yazi.htm **(Ziyaret tarihi:21 Ekim 2006).**

[18] Göçmen, O., 2006, Resim Dosyası İçerisine Steganografik Yöntemlerle Bilgi Gizleme, http://www.baskent.edu.tr/~ogul ogul@baskent.edu.tr **(Ziyaret tarihi:21 Eylül 2006).**

[19] Akar, F., "Veri Gizleme ve Şifreleme Tabanlı Bilgi Güvenliği Uygulaması" , Doktora Tezi, ***Marmara Üniversitesi Fen Bilimleri Enstitüsü***, 21–43, (2005).

[20] Amin M. M., Salleh M., Ibrahim S., and Katmin M. R., "Steganografik: Random LSB Insertion Using Discrete Logarithm," ***Proceedings of 3rd International Conference on Information Technology in Asia*** (CTA03), pp. 234–238, (2003).

[21] Nevit, 2006, Görüntü Dosya Formatları, http://www.fotografim.com/ modules.php?name=Reviews&rop=showcontent&id=5 **(Ziyaret tarihi: 25 Kasım 2006).**

[22] Craver, S., Memon, N., Yeo, B., Yeung, N.M., ***Resolving Rightful Ownerships with Invisible Damgalama Techniques, Research Report RC 20755 (91985)***, Computer Science/Mathematics, IBM Research Division, (1997).

[23] Franz, E., Jerichow, A., Moller, S., Pfitzmann, A., Stierand, I., ***Computer Based Steganografik: How it works and why therefore any restrictions on cryptography are nonsense, at best, Proc. Information Hiding Workshop***, pp. 7–21, (1996).

[24] Johnson, N.F., Jajodia, S., ***Steganalysis of Images Created Using Current Steganografik Software, Proc. Information Hiding Workshop***, Portland, Oregon, USA, (1998).

[25] Fabien, A., Petitcolas, P., Ross J. A. and Markus,G. K, Information Hiding–A Survey, Proceedings of the IEEE, ***Special İssue On Protection Of Multimedia Content***, 87(7), 1062–1078, (1999).

[26] İletim Denetimi Protokolü (TCP), 2006, mikrosoft, http://www.microsoft.com/technet/prodtechnol/windowsserver2003/tr/library/ServerHelp/14fe8097–2828–4d19–bc04–96914553e184.mspx?mfr=true **(Ziyaret tarihi:3 Ekim 2006).**

EK–A: RESİM İÇERİSİNE VERİ GİZLEME UYGULAMASINA AİT PROGRAM KODLARI

Bu ekte, resim içerisine veri gizleme uygulamasının java programlama dilinde yazılan kodları verilmektedir.

```
public  File Yaz(String Mesaj, String DosyaAdi, File EskiDosya)
   {
          FileOutputStream DosyaStream;
          String strMesaj = Mesaj;
               String strMesajUzunluk;
               String strMesajUzunlukBytes ="";
               File fDosya = EskiDosya;
               byte[] bDosya;
               String strYeniMesajBytes;
               String strMesajBytes;
               char[] achMesajBytes;
               byte x,y;
               int z,k,j;
               int basla=54;
               File YeniDosya = new File(DosyaAdi);
          byte[] m={0,1,2};
          int rgb=0;
          byte[] strMesajBytes1 = strMesaj.getBytes();
          long startTime = System.nanoTime();

          try {
            //LSB YÖNTEM SEÇILMISSE:
              if(yontem1.isSelected())
              {
                        DosyaStream = new FileOutputStream(DosyaAdi);
                        strMesajUzunluk = Integer.toString(strMesaj.length(),2);
                        System.out.println("strMesaj:" + strMesaj);
                        System.out.println("strMesajUuzunluk:" + strMesajUzunluk);

                        switch (strMesajUzunluk.length())
                        {
      case 0 : strMesajUzunlukBytes = "00000000";
      case 1 : strMesajUzunlukBytes = "0000000" + strMesajUzunluk;break;
      case 2 : strMesajUzunlukBytes = "000000" + strMesajUzunluk;break;
      case 3 : strMesajUzunlukBytes = "00000" + strMesajUzunluk;break;
      case 4 : strMesajUzunlukBytes = "0000" + strMesajUzunluk;break;
```

```
case 5 : strMesajUzunlukBytes = "000" + strMesajUzunluk;break;
case 6 : strMesajUzunlukBytes = "00" + strMesajUzunluk;break;
case 7 : strMesajUzunlukBytes = "0" + strMesajUzunluk;break;
case 8 : strMesajUzunlukBytes = strMesajUzunluk;break;
default: System.out.println("Mesaj çok uzun.");break;
                }
System.out.println("strMesajUzunlukBytes:" + strMesajUzunlukBytes);
        bDosya = getBytesFromFile(fDosya);
        strMesajBytes = ascii2Bin(strMesaj);
        strYeniMesajBytes = strMesajUzunlukBytes + strMesajBytes;
        achMesajBytes = strYeniMesajBytes.toCharArray();

                if (!(bDosya.length>=strMesaj.length()*8+62))
                {
                        System.out.println("mesaj uygun degil");
                }
                else
                {
        for (int i=0;i<achMesajBytes.length;i++)
        {
                if (achMesajBytes[i]=='0')
                        {
                        x = bDosya[basla+i];
                        y = (byte) (bDosya[basla+i] & 1);
                        z = (int) y;
                        if (z==1)
                        {
                        k = (int) bDosya[basla+i];
                        k--;
                        bDosya[basla+i] = (byte) k;

                        }
        }
              if (achMesajBytes[i]=='1')
                        {
                         x = bDosya[basla+i];
                         y = (byte) (bDosya[basla+i] &1);
                         z = (int) y;
                        if (z!=1)
                        {
                        k = (int) bDosya[basla+i];
                        k++;
                        bDosya[basla+i] = (byte) k;
                                }
                        }
           }//for
                DosyaStream.write(bDosya);
                DosyaStream.close();
```

```
                }//else
        }
        //RGB ASCII YÖNTEMI SEÇILMISSE;
        else if(yontem2.isSelected())
        {
           DosyaStream = new FileOutputStream(DosyaAdi);
           strMesajUzunluk = Integer.toString(strMesaj.length(),2);
           System.out.println("strMesaj:" + strMesaj);
           System.out.println("strMesajUuzunluk:" + strMesajUzunluk);
           switch (strMesajUzunluk.length())
           {
case 0 : strMesajUzunlukBytes = "00000000";
case 1 : strMesajUzunlukBytes = "0000000" + strMesajUzunluk;break;
case 2 : strMesajUzunlukBytes = "000000" + strMesajUzunluk;break;
case 3 : strMesajUzunlukBytes = "00000" + strMesajUzunluk;break;
case 4 : strMesajUzunlukBytes = "0000" + strMesajUzunluk;break;
case 5 : strMesajUzunlukBytes = "000" + strMesajUzunluk;break;
case 6 : strMesajUzunlukBytes = "00" + strMesajUzunluk;break;
case 7 : strMesajUzunlukBytes = "0" + strMesajUzunluk;break;
case 8 : strMesajUzunlukBytes = strMesajUzunluk;break;
default: System.out.println("Mesaj çok uzun.");break;
                }
System.out.println("strMesajUzunlukBytes:" + strMesajUzunlukBytes);
bDosya = getBytesFromFile(fDosya);
System.out.println("Mesajin Boyu.....:"+strMesaj.length());
System.out.println("Resimin Boyu.....:"+bDosya.length);

if (!(bDosya.length>=strMesaj.length()*3+55))
                        {
System.out.println("mesajin boyu resim için uygun degil");
        }
        else
        {
                bDosya[basla+rgb]=(byte)(strMesaj.length());
                rgb++;
                for (int i=0;i<strMesajBytes1.length;i++)
                {
                m[0]=(byte)(strMesajBytes1[i]%10);
                m[1]=(byte)((strMesajBytes1[i]%100)/10);
                m[2]=(byte)((strMesajBytes1[i]-(m[1]*10+m[0]))/100);
                        for(j=0;j<3;j++)
                        {
                        x = bDosya[basla+rgb];
                        if (((int)x>249)&&(m[2-j]<5))         x=(byte)(x-10);
                        x-=(x%10);
                        x+=(10-m[2-j]);
                        bDosya[basla+rgb]=x;
                        rgb++;
                        }
```

```
                        }
                        //yazma işlemi için geçen süreyi hesaplıyor
  long estimatedTime = System.nanoTime() - startTime;
 System.out.println(" yazma islemi icin gecen süre:" + estimatedTime/1000+" mili sn
ecmis");
                DosyaStream.write(bDosya);
                DosyaStream.close();
                }//else
            }
          //IKIBIT YÖNTEMI SEÇILMISSE;
             else if(yontem3.isSelected())
             {
               DosyaStream = new FileOutputStream(DosyaAdi);
               strMesajUzunluk = Integer.toString(strMesaj.length(),2);
               System.out.println("strMesaj:" + strMesaj);
               System.out.println("strMesajUuzunluk:" + strMesajUzunluk);
                switch (strMesajUzunluk.length())
                   {
          case 0 : strMesajUzunlukBytes = "00000000";
          case 1 : strMesajUzunlukBytes = "0000000" + strMesajUzunluk;break;
          case 2 : strMesajUzunlukBytes = "000000" + strMesajUzunluk;break;
          case 3 : strMesajUzunlukBytes = "00000" + strMesajUzunluk;break;
          case 4 : strMesajUzunlukBytes = "0000" + strMesajUzunluk;break;
          case 5 : strMesajUzunlukBytes = "000" + strMesajUzunluk;break;
          case 6 : strMesajUzunlukBytes = "00" + strMesajUzunluk;break;
          case 7 : strMesajUzunlukBytes = "0" + strMesajUzunluk;break;
          case 8 : strMesajUzunlukBytes = strMesajUzunluk;break;
          default: System.out.println("Mesaj çok uzun.");break;
                        }
          System.out.println("strMesajUzunlukBytes:" + strMesajUzunlukBytes);
          bDosya = getBytesFromFile(fDosya);
          strMesajBytes = ascii2Bin(strMesaj);
          strYeniMesajBytes = strMesajUzunlukBytes + strMesajBytes;
          achMesajBytes = strYeniMesajBytes.toCharArray();
          if (!(bDosya.length>=strMesaj.length()*4+58))
                  {
                          System.out.println("mesaj uygun degil");
                  }
                  else
                  {
                       int m1=0;
                  for (int i=0;i<achMesajBytes.length;i=i+2)
                  {
                          if (achMesajBytes[i]=='0')
                          {
                                  x = bDosya[basla+m1];
                                  y = (byte) (bDosya[basla+m1] & 1 );
                                  z = (int) y;
                                  if (z==1)
```

```
                {
                        k = (int) bDosya[basla+m1];
                        k--;
                        bDosya[basla+m1] = (byte) k;
                                }
                }
                    if (achMesajBytes[i]=='1')
                        {
                         x = bDosya[basla+m1];
                         y = (byte) (bDosya[basla+m1] &1);
                         z = (int) y;
                          if (z!=1)
                        {
                        k = (int) bDosya[basla+m1];
                        k++;
                        bDosya[basla+m1] = (byte) k;
                        }
                        }
                        if (achMesajBytes[i+1]=='0')
                        {
                        y = (byte) (bDosya[basla+m1] & 2);
                        z = (int) y;
                        if (z==2)
                                {
                        k = (int) bDosya[basla+m1];
                        k=k-2;
                        bDosya[basla+m1] = (byte) k
                        }
                        }
                        if (achMesajBytes[i+1]=='1')
                        {
                         x = bDosya[basla+m1];
                        y = (byte) (bDosya[basla+m1] & 2);
                        z = (int) y;
                        if (z!=2)
                        {
                        k = (int) bDosya[basla+m1];
                        k=k+2;
                        bDosya[basla+m1] = (byte) k;
                        }
                        }
                m1++;
        }//for
                DosyaStream.write(bDosya);
                DosyaStream.close();
                }
}
else
{
```

```
                    return YeniDosya;
            }
            } //try
                catch(java.io.FileNotFoundException e)
                    {
                        System.out.println(e.getMessage());
                    }
                    catch(java.io.IOException e)
                    {
                            System.out.println(e.getMessage());
                    }
                    return YeniDosya;
    }  // Yaz Yöntemi
}// Sınıf Sonu
```

EK–B: RESİM İÇERİSİNDEKİ GİZLİ VERİYİ ÇIKARMA YGULAMASINA AİT PROGRAM KODLARI

Bu ekte, resim içerisindeki gizli veriyi çıkarma uygulamasının java programlama dilinde yazılan kodları verilmektedir.

```
private String Okuyucu(File file)
{
        String Sonuc="";
        try {

byte[] bDizi = getBytesFromFile(file);
StringBuffer bfrKacKarakter = new StringBuffer();
StringBuffer bfrMesaj = new StringBuffer();

int temp ;
// LSB YÖNTEMİ SEÇİLMİŞSE

if(yontem1.isSelected())
{ for (int i=54;i<=61;i++)
 {
  temp = bDizi[i];
  bfrKacKarakter.append(temp & 1);
 }
int MesajUzunluk = Integer.parseInt( bfrKacKarakter.toString(),2 );
 for (int j=62;j<62+MesajUzunluk*8;j++)
 {
  temp = bDizi[j];
  bfrMesaj.append(temp & 1);

 }
 Sonuc = bin2Ascii( bfrMesaj.toString() );

// RGB–ASCII YÖNTEMİ SEÇİLMİŞSE
}else if(yontem2.isSelected())
{
byte[] strMesajBytes= {0,1,2,3,4,5,5,5,5,5,5,12,10,1,1,5,12,10,1,1,5,12,10,1,1};
int m1,m2,m3,j=0 ;
//mesajın boyunu okumak için
byte mesajBoyutu = (byte)bDizi[54];
//mesajin kendisin okumak için
for (int s=55;s<55+(mesajBoyutu*3);s+=3)
```

```
 {
  m3=(10–(bDizi[s]%10))%10;
  m2=(10–(bDizi[s+1]%10))%10;
  m1=(10–(bDizi[s+2]%10))%10;
  strMesajBytes[j]=(byte)(m3*100+m2*10+m1);
  bfrMesaj.append((char)strMesajBytes[j]);
 }
  log.append(satirBasi+"Okunan Mesaj : "+bfrMesaj);

// İKİ BİT YÖNTEMİ SEÇİLMİŞSE
 }else if(yontem3.isSelected())
 {
 int temp2;
 int temp3;
 //mesajın boyunu okumak için
 for (int i=54;i<=57;i++)
 {
  temp = bDizi[i];
  bfrKacKarakter.append(temp & 1);
  temp2=(bDizi[i]&2);
  if (temp2==2)
  {
   bfrKacKarakter.append(1);
  }
  if (temp2!=2)
  {
   bfrKacKarakter.append(0);
  }
 }
 int MesajUzunluk = Integer.parseInt( bfrKacKarakter.toString(),2 );
 //mesajin kendisin okumak için
 for (int s=58;s<= MesajUzunluk*4+58;s++)
 {
  bfrMesaj.append(Integer.toBinaryString(bDizi[s] & 1));
  System.out.println(bfrMesaj.toString());
  temp2=(bDizi[s]&2);
  if (temp2==2)
   bfrMesaj.append(Integer.toBinaryString(1));
  else
   bfrMesaj.append(Integer.toBinaryString(0));
 }
 Sonuc = bin2Ascii(bfrMesaj.toString() );
 }
}catch(java.io.IOException e)
{   }
return Sonuc;
}
```

ÖZGEÇMİŞ

1979 yılında Iğdır'ın Kacer Doğan Şanlı köyünde doğdu. İlk, orta ve lise öğrenimini İstanbul'da tamamladı. 1997 yılında girdiği Kocaeli Üniversitesi Teknik Eğitim Fakültesi Elektronik ve Bilgisayar Eğitimi Bölümü Bilgisayar Öğretmenliği Programından 2002 yılında Bilgisayar Teknik Öğretmeni olarak mezun oldu. 2004 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Elektronik Bilgisayar Eğitimi Anabilim Dalı'ndaki yüksek lisans eğitimine halen devam etmektedir. 2002–2005 yılları arasında Körfez Milangaz Hacer Demirören Çok Programlı Lisesi'nde bilgisayar öğretmeni olarak görev yaptı. 2006 yılından beri Kartal Vali Erol Çakır Ticaret Meslek Lisesi'nde bilgisayar öğretmeni olarak çalışmaktadır.