T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VERİ GİZLEMEDE SIKIŞTIRMA ALGORİTMASI KULLANIMI VE UYGULAMASI

YÜKSEK LİSANS TEZİ

Esra Ayça GÜZELDERELİ

Enstitü Anabilim Dalı : ELEKTRONİK VE BİLG. EĞT.

Tez Danışmanı : Doç. Dr. Özdemir ÇETİN

Mayıs 2012

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

VERİ GİZLEMEDE SIKIŞTIRMA ALGORİTMASI KULLANIMI VE UYGULAMASI

YÜKSEK LİSANS TEZİ

Esra Ayça GÜZELDERELİ

Enstitü Anabilim Dalı : ELEKTRONİK VE BİLG. EĞT.

Bu tez 25 / 05 /2012 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof.Dr.	**Yrd.Doç.Dr.**	**Doç.Dr.**
Hüseyin EKİZ	**Murat İSKEFİYELİ**	**Özdemir ÇETİN**
Jüri Başkanı	**Jüri Üyesi**	**Jüri Üyesi**

TEŞEKKÜR

Bu çalışmanın hazırlanması esnasında bana yol gösteren, bu alanda çalışmam için beni teşvik eden, yardımlarını ve desteğini benden esirgemeyen değerli danışman hocam Doç. Dr. Özdemir ÇETİN' e teşekkür ederim.

Çalışmalarım sırasında çalışabilmem için gerekli ortamı sağlayan, manevi desteğini benden esirgemeyen ve her zaman yanımda olan başta ablam Azer GÜZELDERELİ olmak üzere sevgili aileme teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

ASCII	: American Standard Code for Information Interchange
BMP	: Bit Map
CCITT	: Consultive Committee on International Telephone Telegraph
DCT	: Ayrık kosinis dönüşümü
DFT	: Ayrık Fourier dönüşümü
DWT	: Ayrık dalgacık dönüşümü
GIF	: Graphics Interchange Format
İDS	: İnsan Duyma Sistemi
ISSDC	: Tekrarlı Yarı-Statik Diagram Kodlaması (Iterative Semi-Static Diagram Coding)
JPEG	: Joint Photographic Experts Group
LSB	: En Az Ağırlıklı Bit (LSB, Least Significant Bit)
LZ77	: Lempel Ziv 77
LZ78	: Lempel Ziv 78
LZSS	: Lempel Ziv Storer Szymanski
LZW	: Lempel Ziv Welch
MPEG	: Motion Picture Expert Group
RGB	: Kırmızı Yeşil Mavi (RGB, Red Green Blue)
SSDC	: Yarı-Statik Diagram Kodlaması (Semi-Static Diagram Coding)
YCbCr	: Luma blue-difference, red-difference

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Steganografi, Veri Gizleme, Kısmi Optimizasyon, LZW Sıkıştırma

Günümüzde steganografi(veri gizleme) uygulamaları giderek artan yaygın bir öneme sahip olmaktadır. Steganografi üzerine yapılan çalışmalar, çoklu ortam ve bilgi güvenliği uygulamaları gibi güncel gereksinimler ile daha da artan bir talep görmektedir. Steganografi' nin amacı gizli mesaj ya da bilginin varlığını saklamaktır. Gizlenmek istenen mesaj, bir başka masum görünüşlü ortamda saklanır ve böylece üçüncü şahısların gizlenen mesajın varlığından haberdar olması engellenir. Bu açıdan özellikle resim içerisine bilgi gömme teknikleri üzerine yoğunlaşarak, bu alanda yeni yaklaşımlar elde edilmiştir. Bu yaklaşımlardan biri de kısmi optimizasyon yöntemi ile veri gizleme tekniğidir.

Kısmi optimizasyon tekniğinde resmi oluşturan tüm piksel hücreleri sekiz farklı bölgede sınıflandırılarak, her bölgeye sekiz farklı optimizasyon uygulanmaktadır. Böylelikle her bir bölgeden en efektif sonuç elde edilerek, değiştirilen bit (hatalı bit) sayısı minimuma indirgenmektedir. Bu tez çalışmasında, kısmi optimizasyon tekniğinin bir sıkıştırma algoritmasıyla desteklenmesiyle, veri gizleme işleminin daha efektif sonuçlar vermesi sağlanmıştır. Bunun için gizlenecek olan verinin kayıpsız ve dinamik sözlük tabanlı bir sıkıştırma tekniği olan LZW sıkıştırma algoritması ile sıkıştırılarak, boyutunun minimize edilmesi sağlanmıştır. Bu sayede gizleme işlemi sonucu resimde meydana gelen bozulma en aza indirgenmiş ve veri gizleme işleminin daha kısa süre içerisinde gerçekleştirilmesi sağlanmıştır.

DATA HIDING APPLICATION BASED ON A NEW COMPRESSING ALGORITHM

SUMMARY

Key words : Steganography, Data Hiding, Partial Optimization, Lzw compression

Recently, the applications of data hiding (steganography) have been more common and had an increasing importance. The studies on data hiding have been demanded much more with daily needs such as multimedia and information security. The aim of steganography is to hide any secret message or data. The message that is required to be sent is prevented by being hided in a innocent-looking environment so that a third person can't be aware of the existence of the message. This aspect has leaded to many new approaches about this subject by focusing on the techniques used for data hiding, especially hiding data in a picture. One of these approaches is the Data Hiding with Partial Optimization Method.

In Data Hiding with Partial Optimization Method, all the picsels that form the Picture are put into categories in eight different areas and eight different partial optimization is applied to each area. So that the number of bit which was changed before is reduced to minimum by obtaining the most effective results from each area. With this thesis study, it is provided to get more effective results from data hiding by supporting the Partial Optimization technique with a compression algorithm. For this aim, the data is compressed by a technique, called LZW compression algorithm and it is provided to minimize its size. This way the defects that appeared in the Picture as a result of data hiding are reduced to minimum and the process of data hiding takes shorter time.

BÖLÜM 1. GİRİŞ

Gün geçtikçe artan veri trafiği ve buna paralel artan veri hırsızlığı sebebiyle önemli verileri gizlemek ve bunları güvenli bir ortamda saklamak oldukça zor hale gelmiştir. Bilgi güvenliği, günümüzde kişisel bazdaki öneminden çok, bazı toplumların geleceklerinin teminatı olan özel iş ve görev yapan birimlerde/kurumlarda önem arz etmektedir. Silahlı kuvvetler buna en iyi örneklerden birisi olabilir. Günümüzde gelişen teknoloji ile birlikte sayısal ortamdaki (metin, ses ve görüntü dosyaları) verilerin korunma ihtiyacı fazlasıyla ortaya çıkmaktadır.

Bilişim teknolojilerinin hayatımıza daha fazla girmesi ve yaygınlaşmasıyla birlikte, yapılan iş ve işlemler elektronik ortamlara kaymakta, bu ortamlarda bulunan, saklanan, işlenen ve transfer edilen bilgilerin ise korunması veya güvenliğinin sağlanması çok büyük önem arz etmektedir.

Sayısal olarak veri iletişimi gerçekleştirilen bir ortamda, göndericiden alıcıya giden veriye yönelik izinsiz erişim, zarar verme, yok etme, değişiklik yapma ve yeniden üretme gibi birçok tehdit mevcuttur. Bu tehditlerin alınan önlemlere rağmen her geçen gün arttığı görülmektedir. Bu tehditlerin ortaya çıkmasına karşılık olarak, bunların ortadan kaldırılması için de çeşitli teknikler geliştirilmiştir.

Steganografinin temel mantığı, sayısal veri dosyası formatlarındaki gereksiz veya çok önemli olmayan kısımların veya insan duyularının zayıf kaldığı ya da sınırlarının kullanılmasına dayanmaktadır.

Tam olarak “kaplanmış yazı” anlamına gelen steganografinin amacı, önemli mesaj ya da bilginin varlığını saklamaktır. Steganografi kelimesi, Yunanca “steganos: gizli, saklı” ve “grafi: çizim ya da yazım” kelimelerinden gelmektedir [1]. Verilerin korunmasında ve gizlenmesinde steganografi önemli rol oynamaktadır. Eskiden

ilkel yöntemlerle yapılmaya çalışılan steganografi, günümüzde yazılım ile desteklenmekte ve güçlü algoritmalar kullanılmaktadır. Bilgi gizlemenin önemli bir alt disiplini olan steganografi, bir nesnenin içerisine bir verinin gizlenmesi olarak tanımlanabilir. Steganografinin amacı gizli mesaj ya da bilginin varlığını saklamaktır. Taşınmak istenen mesaj bir başka masum görünüşlü ortamda saklanarak, üçüncü şahısların iletilen mesajın varlığından haberdar olması engellenir [2]. Bu açıdan özellikle resim içerisine bilgi gömme teknikleri üzerine yoğunlaşarak, bu alanda yeni yaklaşımlar elde edilmiştir [3]. Bu yaklaşımlardan biri de 'Kısmi Optimizasyon Yöntemi ile Veri Gömme Tekniği'dir.

Kısmi optimizasyon yöntemi ile veri gömme tekniğinde, gömme objesi olarak resim kullanılmıştır. Resim, sekiz farklı optimizasyon bölgesine ayrılarak, her bir bölge sekiz farklı optimizasyon işlemine tabi tutulmaktadır. Optimizasyon kodlaması ile bir çeşit bölgesel kodlama yapıldığından, her bir bölgede veri gömme sırası değişmekte olup 256 farklı kodlama türünün oluşması sağlanmaktadır. Her bir bölgeden elde edilen sonuçlar en efektif değerler olup, buna göre değiştirilen (hata) bitlerin en aza indirgendiği sonucuna ulaşılmaktadır. Bu durumda her bir bölgeye farklı metot uygulanacağından bölgelerarası farklı kodlama teknikleri sebebiyle hatalı kod çözme işlemi en aza indirgenecektir. Dolayısıyla bu yöntemle, minimum hata oranı için korunan orijinal piksellerin sayısı artırılmakta, ayrıca kendi içerisinde güvenlik sağlanmaktadır [4]. Kısmi optimizasyon yöntemi ile veri gömme tekniğinin sıkıştırma algoritmaları ile desteklenerek daha iyi sonuçlar elde edeceği öngörülmektedir.

Veri sıkıştırma, verilerin hafıza alanında daha az yer işgal etmeleri ve bir iletişim ağı üzerinden daha hızlı transfer edilebilmeleri için yaygın olarak kullanılmakta olan tekniklerdir. Bu sayede bellek üzerinde yer tasarrufu, veri aktarımında da zaman tasarrufu yapılabilmektedir. Veri sıkıştırma yöntemleri farklı temellere, farklı tipte verilere göre farklı sonuçlar üretse de temelde hepsi "gereksiz verileri yok etme" prensibine dayanmaktadır. Son yıllarda disk kapasitelerinin hızlı bir şekilde artması, genel amaçlı sıkıştırma uygulamalarının kullanım oranını azaltmıştır. Fakat sabit disklerimizde sakladığımız ses, görüntü ve hareketli görüntü dosyalarının tamamına yakını çeşitli yöntemlerle sıkıştırılmış haldedir [5].

Veri sıkıştırma yöntemleri, kayıplı ve kayıpsız sıkıştırma yöntemleri olmak üzere ikiye ayrılır. Kayıplı sıkıştırmada, verinin bütünlüğünü en az düzeyde etkileyecek olan veri kümeleri çıkartılır, geriye kalan veri kümeleri kayıpsız sıkıştırmaya tâbi tutularak sıkıştırma sağlanır. Bir veri kayıplı bir sıkıştırma yöntemi ile sıkıştırılırsa, verinin tamamı değil, sadece belirli bir kısmı geri getirilebilir. Kayıpsız sıkıştırma yöntemleri ise orijinal veri ile sıkıştırıldıktan sonra geri getirilecek olan verinin tamamıyla aynı olmasının gerekli olduğu durumlarda kullanılır. İnsan gözünün ve kulağının hassasiyeti ile direkt olarak ilgisi bulunmayan, metin belgeleri, kaynak kodları, çalıştırılabilir program dosyaları gibi dosyalar kayıpsız sıkıştırılmak zorundadırlar. Kayıpsız sıkıştırma tekniklerinden en bilineni, dinamik sözlük yaklaşımı temeline dayanan, LZ ailesinin en çok bilinen ve en iyi sıkıştırma oranı sağlayan üyesi, 1984 yılında Terry Welch tarafından yayınlanan LZW algoritmasıdır.

Terry Welch 1984'te Unisys için çalışırken, LZ78 yaklaşımını yüksek performanslı disk ünitelerine uyarlamış ve ortaya çıkan yeni algoritma LZW olarak kabul görmüştür. LZW hem sıkıştırma hem de açma performansı açısından LZ78 ailesinin en iyisi olmayı başarmıştır. Her tip veri üzerinde iyi sonuçlar veren bir algoritma olduğu için, kendisinden sonra gelen birçok algoritma, LZW' yi temel almıştır.

Bu tez çalışmasında kısmi optimizasyon tekniğinin bir sıkıştırma algoritmasıyla desteklenmesiyle, veri gizleme işleminden daha efektif sonuçlar elde edilmesi sağlanmıştır. Bunun için optimizasyon ile gizlenecek olan verinin, kayıpsız ve dinamik sözlük tabanlı bir sıkıştırma tekniği olan LZW sıkıştırma algoritması ile sıkıştırılarak, boyutunun minimize edilmesi sağlanmıştır. Bu sayede optimizasyon ile veri gizleme işlemi sonucu resimde meydana gelen bozulma en aza indirgenmiş ve veri gizleme işleminin daha kısa süre içerisinde gerçekleştirilmesi sağlanmıştır.

Bu tez çalışması beş bölümden oluşmaktadır. Birinci bölümde bu çalışmanın gereği açıklanmıştır. İkinci bölümde veri gizleme tekniklerinin amacı, kullanım alanları, tarihçesi ve bu alanda kullanılan teknikler hakkında temel bilgiler verilmiştir. Ayrıca görüntü objesi üzerinde gömme uygulamalarında günümüzde kullanılan yöntemler açıklanmıştır. Üçüncü bölümde veri gizleme/gömme teknikleri içerisinde yer alan kısmi optimizasyon tekniği ayrıntılı olarak incelenmiştir. Tekniğin işlem süreçleri,

avantaj ve dezavantajları ve tekniğin uygulanmasıyla elde edilen sonuçlar bu bölümde yer almıştır. Dördüncü bölümde ise veri sıkıştırma teknikleri üzerinde durulmuş ve günümüzde kullanılan teknikler incelenmiştir. Ayrıca tez çalışmasında kullanılan LZW sıkıştırma algoritması da bu bölümde detaylandırılmaktadır. Beşinci bölümde tez çalışmasının temelini oluşturan sıkıştırma algoritması tabanlı veri gizleme uygulaması ile resim dosyası içerisine veri gömme uygulamaları sunulmakta ve yapılan deneysel uygulamadan elde edilen sonuçlar değerlendirilmektedir. Son bölümde ise sonuç ve önerilere yer verilmiştir.

BÖLÜM 2. STEGANOGRAFİ

2.1. Giriş

Bu bölümde; steganografi (veri gizleme) teknikleri hakkında temel bilgilerin verilmesi ile yapılan tez çalışmasının daha iyi anlaşılabilmesi amaçlanmaktadır.

Bilişim teknolojilerinin hayatımıza daha fazla girmesi ve yaygınlaşmasıyla birlikte, yapılan iş ve işlemler elektronik ortamlara kaymakta, bu ortamlarda bulunan, saklanan, işlenen ve transfer edilen bilgilerin ise korunması veya güvenliğinin sağlanması çok büyük önem arz etmektedir.

Sayısal olarak veri iletişimi gerçekleştirilen bir ortamda, göndericiden alıcıya giden veriye yönelik izinsiz erişim, zarar verme, yok etme, değişiklik yapma ve yeniden üretme gibi birçok tehdit mevcuttur. Bu tehditlerin alınan önlemlere rağmen her geçen gün arttığı görülmektedir. Bu tehditlerin ortaya çıkmasına karşılık olarak, bunların ortadan kaldırılması için çeşitli teknikler geliştirilmiştir.

Bu bölümde öncelikle steganografi bilimiyle ilgili temel konular ve tarihçesi ele alınmış, steganografi konusuna genel bir bakış yapılmasının ardından, steganografik yöntemler detaylı olarak incelenmiştir.

Günümüz teknolojisinin gelişmesiyle sayısal ortamda kritik verilerin gizliliği büyük önem kazanmıştır. Terimsel olarak “kaplanmış yazı” anlamına gelen Steganografi’ nin amacı, bilginin varlığını saklamaktır. Steganografi kelimesi, Yunanca “steganos: gizli, saklı” ve “grafi: çizim ya da yazım” kelimelerinden gelmektedir [1].

Bilgi gizlemenin önemli bir alt disiplini olan Steganografi, bir nesnenin içerisine bir başka verinin gizlenmesi olarak tanımlanır. Taşınmak istenen mesajın başka bir ortamda saklanması, böylece üçüncü şahısların iletilen mesajın varlığından haberdar olmaması sağlanır [2].

Steganografi, temel mantığı olarak binlerce yıldır uygulanan bir gizli iletişim yöntemi olmakla birlikte, günümüzde sayısal dosya çeşitliliğinin artmasıyla birlikte çok farklı veri saklama yöntemleri geliştirilmeye başlanmıştır. Steganografi'nin temelinde insan duyularının algılayamayacağı sınırlar içerisinde sayısal veri dosyalarının gizli mesajı içerecek şekilde modifikasyonu yatmaktadır. Örneğin görme sistemi, bir resimdeki çok kücük değişimleri fark edemez. İşitme sistemi de belli bir seviyenin altındaki sesleri işitemez ve görme sistemine göre çok daha hassas olsa bile sesteki küçük değişimleri fark edemez. Steganografi ise görme ve işitme sistemlerinin zayıflıklarını kullanarak, resim ve ses dosyaları içerisine veri saklanmasını desteklemektedir.

2.2. Steganografi'nin Tarihçesi

Steganografi kökleri binlerce yıl öncesine dayanan bir bilim dalıdır [6]. Günümüzde steganografi, sayısal verinin saklanma teknigi olarak geniş bir alanda bilinmesine karşın steganografi, eski zamanlara kadar uzanır. Örneğin eski yunanda M.Ö. 5.yüzyılda Susa kralı Darius tarafından göz hapsine alınan Histiaeus Milet'teki oğlu Aristagoras'a gizli bir mesaj göndermek ister. Histiaeus kölelerinden birinin saçlarını kazıtır ve mesajını dövme şeklinde kölenin kafasına işletir. Kölenin saçları tekrar uzayınca onu Milet'e oğlunun yanına gönderir. Bu gizli yazma sanatı steganografi'nin ilk kullanıldığı yerlerden biridir. Bir başka örnek ise; eski yunanlıların gizli mesajı tabletlere yazıp, sonrasında tabletleri bal mumu ile kaplamalarıdır. Böylece güvenli bir şekilde iletilmesi gereken bilgi gizlenmiş olur. Mesajı alanlar daha sonra bal mumunu kazıyarak gizli bilgiye ulaşırlar [7]. Daha sonraki zamanlarda steganografi, harflere müzik notalarının atanması, I. ve II. Dünya Savaşlarında kullanılan mors kodları, II. Dünya savaşı esnasında başarıyla uygulanan görünmez mürekkeplerin kullanımı gibi uygulamalarla karşımıza çıkmaktadır [8]. Özellikle 11 Eylül sonrası teröristlerce de gizli mesajlaşma için kullanıldığı düşünülen, herhangi bir obje

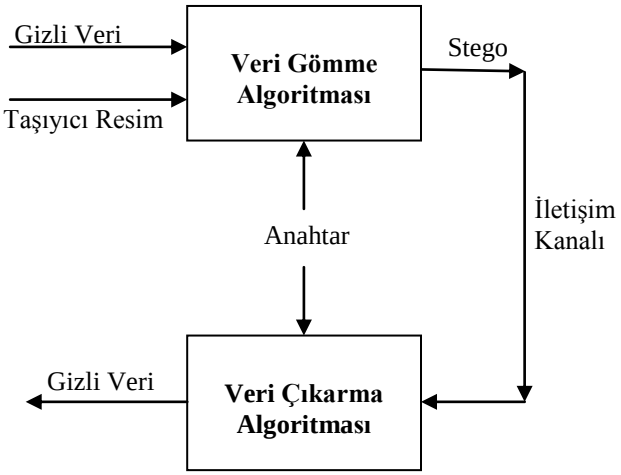
içerisine özelliklerini bozmadan başka bir verinin gizlenmesi mantığına dayanan veri gizleme tekniği artan bir ilgi konusu olmuştur.

2.3. Steganografi'nin Amacı

Bilgi güvenliği günümüzde kişisel olarak önemli olsa da, daha önemlisi bazı toplumların geleceklerinin teminatı olan özel iş ve görev yapan birimler/kurumlar için çok daha fazla önem arz etmektedir. Askeri uygulamalar bunların başında gelmektedir. Bu gibi alanlarda gizliliği sağlamak amacıyla steganografiye sıkça başvurulur.

Günümüz teknolojisinin gelişmesiyle sayısal ortamdaki verilerin gizliliği büyük önem kazanmıştır. İletilecek metnin şifrelenmesi için kullanılan iki temel yaklaşımdan ilki metnin kendi içierisinde karıştırılarak anlaşılmaz hale getirilmesi (kriptografi), ikincisi ise metnin bir başka yapı içerisine gömülerek gizlenmesidir. İkincisi için kullanılan yöntemlerden birisi de steganografidir [9].

Steganografinin en temel amacı, iletişimin gizliliğini sağlamaktır. Steganografide, sayısal bir verinin başka bir sayısal veri içerisine, fark edilebilir değişikliklere sebep olmadan saklanması gerçekleştirilmektedir. Örneğin şifrelenmiş bir metin, bir resim dosyasına saklanmakta ve sonuçta oluşan resim dosyası hem fiziksel olarak hem de görsel olarak orijinalinden farklı olmamaktadır. Böylece iki uç arasındaki iletişimi gözetleyenler, arada sadece transfer edilen bir resim görmekte, ama aslında bu resim yoluyla gizli bir mesajlaşma gerçekleştiğinin farkında olmamaktadırlar [10].



Şekil.2.1. Steganografi sistem yapısı

Şekil 2.1'de steganografi tekniğinin genel sistem yapısı verilmiştir. Buradan da anlaşılacağı üzere, potensiyel bir saldırıdan etkilenmemesi gereken gizli verinin, veri gömme algoritması aracılığıyla taşıyıcı resimde ufak değişiklikler yaparak gizliliği sağlanır. Gizlenecek mesaj, düz metin, şifrelenmiş metin, ya da bitler halinde temsil edilebilecek herhangi bir şey olabilir. Gizleme sonucunda elde edilen, gizli veriyi içerisinde barındıran sayısal ortam "stego" olarak adlandırılır. Elde edilen stego, bir anahtar yardımıyla, veri çıkarma algoritmasına tâbi tutulduğunda gizlenmiş veri tekrar elde edilir.

Steganografi uygulamalarında sağlanması gerekli en önemli özellikler; sağlamlık, keşfedilememe, görünmezlik, güvenlik, karmaşıklık ve daha fazla veri gömme kapasitesine ulaşmaktır [11].

2.4. Steganografi'nin Kullanım Alanları

Steganografi, "Dilbilim Steganografi" ve "Teknik Steganografi" olmak üzere kendi içerisinde ikiye ayrılmaktadır. Dilbilim steganografi, taşıyıcının metin olduğu steganografi koludur. Teknik Steganografi ise birçok konuyu içine almaktadır [8].

2.4.1. Dilbilim steganografi

Dilbilim Steganografi, taşıyıcı verinin text olduğu steganografi koludur. Burada değişiklik yapmanın çeşitli yolları vardır. Bunlar; grafik kullanılarak yapılabilir, text' in yapısı değiştirilerek yapılabilir ya da amacı sadece veriyi saklamak olan yeni bir text oluşturulabilir. Dilbilim Steganografi'de kullanılan yöntemler şunlardır:

Açık kodlar: Gizli mesaj, açıkça okunabilir fakat zararsız bir mesaj haline gelir. Bu işlem; maskeleme, boş şifreler ve grid (ızgara) ile yapılmaktadır [12].

Şemagramlar: Gizli mesaj, açık metinin ufak, fakat gizli bir detayının içine gizlenmektedir. Bunun için grafiksel değişiklikler yapılmaktadır. Kullanılan yöntemler ise; farklı yazı tipleri kullanmak, eski daktilo yazılarını kullanmak, resimler içinde boşluklar kullanmak gibi yöntemlerdir [12].

2.4.2. Teknik steganografi

Teknik Steganografi, birçok konuyu içine almaktadır. Bunları bazı başlıklar altında toplayabiliriz;

Görünmez mürekkep: Geleneksel haline gelmiş olan görünmez mürekkeple yazma yöntemidir.

Gizli yerler: Kimsenin göremeyeceği gizli yerlere saklama (bavul, kasa vb.)

Microdot'lar: Bilgiyi noktalar halinde sayfaya gizleme

Bilgisayar tabanlı yöntemler: Text, ses, hareketli görüntü, resim dosyalarını kullanarak veri gizleme yöntemleridir. Bilgisayar tabanlı steganografi, kullanım alanları açısından üçe ayrılmaktadır. Bunlar aşağıdaki gibidir:

- Metin (text) steganografi
- Görüntü (image) steganografi

- Ses (audio) steganografi.

2.4.3. Metin steganografi

Metin steganografi taşıyıcı ortamın text olduğu steganografi alanıdır. Genelde uygulanması zor bir veri gizleme şeklidir. Metin Steganografi'de saklanacak veri miktarı azdır. Bunun nedeni taşıyıcı text'in içindeki gereksiz alanların ve boşlukların miktarının az olmasıdır [13].

Metin tabanlı gizleme yöntemlerinin hepsi, gizli mesajı geri çözebilmek için ya orijinal metne, yada orijinal metnin biçimlendirme bilgisine ihtiyaç duyar.

Metin Steganografi veri saklanacak yerlerin özelliklerine göre aşağıdaki yöntemleri kullanır.

- Açık alan yöntemleri
- Yazımsal yöntemler
- Anlamsal yöntemler

Açık alan yöntemler, iki kelime arasında extra boşluklar, satır sonu boşlukları ile çalışmaktadır. Bununla birlikte, bu yöntemlerinn ASCII kodları ile kullanılması daha uygundur. Açık alan yöntemleri de kendi içerisinde 5 farklı uygulama tipine sahiptir. Bunlar; cümle içi boşluk bırakma, satır kaydırma, satır sonu boşluk bırakma, sağa hizalama, gelecek kodlaması gibi uygulamalardır.

Yazımsal yöntemler, dökümanı kodlamak için noktalama işaretlerini kullanır. Örneğin "ekmek, peynir**,** ve süt" cümlesi ile "ekmek, peynir ve süt" cümlesi ilk bakışta aynıymış gibi gözükmektedir. Fakat dikkatlice bakıldığında ilk cümlenin fazladan bir ',' işareti içerdiği görülür. Bu yapıların biri "1", diğeri de "0" olarak belirlenir ve kodlama işlemi bu şekilde yapılır.

Anlamsal yöntemler ise W. Bender tarafından ortaya atılmıştır. Bu yöntemde eş anlamlı kelimelere birincil ve ikincil değerler atanmaktadır. Sonra bu değerler "1" ve

“0” olarak binary’e dönüştürülür. Örneğin “uzun” kelimesi birincil, “kısa” kelimesi de ikincil olarak işaretlenmiş olsun. Birincil “1”, ikincil de “0” olarak binary’e çevrilir.

2.4.4. Görüntü steganografi

Görüntü dosyaları içerisine saklanacak veriler metin dosyası olabileceği gibi, herhangi bir görüntü içerisine gizlenmiş bir başka görüntü dosyası da olabilir. Bu yaklaşımda içine bilgi gizlenen ortama örtü verisi, oluşan ortama da stegometin veya stego-nesnesi denmektedir [1].

Görüntü steganografide, resim üzerinde gerçekleştirilen değişiklikler insan gözü tarafından algılanmamalıdır. Aksi halde en azından bir gizli metin iletilmekte olduğu anlaşılacaktır. Bu durumda üçüncü kişilerin içerisinde gizli veri olan resim üzerinde işlemler yapma olasılığı yüksektir. Steganografik yöntemlerin bu tür saldırılara karşı bir dayanıklılık göstermesi gerekir. Bugüne kadar yapılan steganografi çalışmaları şu şekilde sınıflandırılabilir [1].

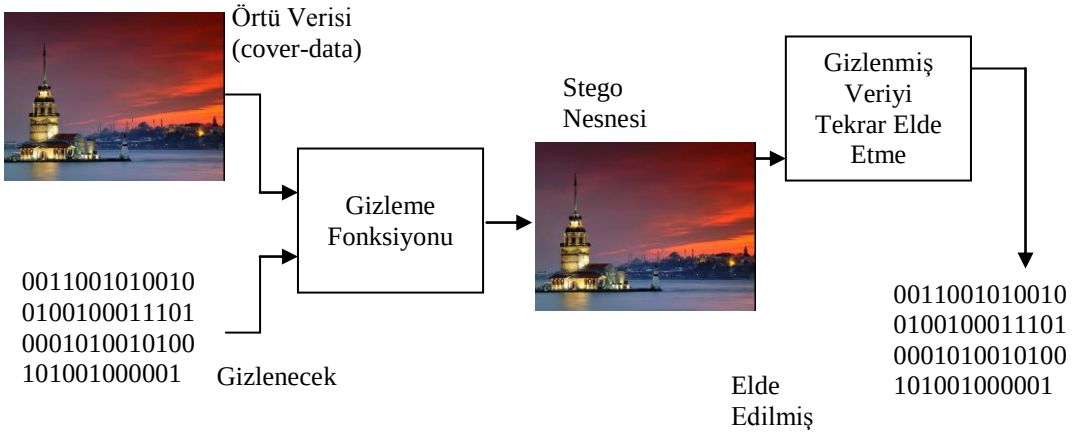
- Yer değiştirmeye dayalı yöntemler
- İşaret işlemeye dayanan yöntemler
- Spektrum yayılmasına dayanan yöntemler
- İstatistiksel yöntemler

Yer değiştirmeye dayalı yöntemlerde, temel olarak resim dosyasında piksel renklerini temsil eden değerler üzerinde çalışılmaktadır. Pikselin rengi bir bayt ile ifade edilmektedir. Bu baytların en düşük bitlerinin değişmesi resim görüntüsünde gözle farkedilmesi zor bir fark yaratmaktadır. Metne ait karakterleri temsil eden byte’ların her bir bitinin farklı bir pikselin en önemsiz bitine kaydedilmesi “en önemsiz bite ekleme yöntemi” (LSB - Least Significant Bit Insertion Methods) olarak bilinir. Sonuçta ortaya çıkan resim dosyasında renk değerleri ya olduğu gibi kalır ya bir artar ya da azalır. Her üç durumda da insan gözü tarafından algılanamamaktadır. Burada veri bit değerleri sırasıyla eklenmektedir.

İşaret işlemeye dayanan yöntemlerde, resim sıkıştırma algoritmaları kullanılmaktadır. Tüm sıkıştırma algoritmaları insan gözünün filtreleme özelliğini kullanmaktadırlar. İnsan gözü belli bir frekanstan sonrasındaki renk değişimlerini algılayamamaktadır. Dolayısıyla, asıl resimdeki renk değerleri frekans düzlemine taşınabilir. Frekans düzleminde çeşitli katsayılar oluşacaktır. Ancak bu katsayılar kullanılarak ters işlem sonrası tekrar asıl resmi elde etmek için sonsuz sayıda frekans bileşeninden faydalanmak gerekmektedir.

Spektrum yayılmasına dayanan yöntemlerde, gönderilmek istenen mesaj ihtiyaç duyduğu frekans bandından çok daha fazlasına dağıtılmaktadır. Üçüncü bir kişi araya girip bir ya da birden fazla frekans bandında bozulmalara neden olsa bile, alıcı geri kalan frekans bantlarındaki bilgiler ile asıl mesajı elde edebilmektedir.

Görüntü dosyaları için bir steganografik sistem Şekil 2.2' de gösterilmektedir. Gönderici bir gizleme fonksiyonu kullanarak bir stego nesnesi oluşturur. Gizleme fonksiyonu, verinin saklanacağı taşıyıcı ortam ve gizlenecek veri olmak üzere iki parametreye sahiptir [8].



Şekil 2.2. Görüntü için steganografik sistem

Görüntü steganografide, bilgilerin görüntü dosyaları içerisine saklanmasının çeşitli yöntemleri vardır. Şekil 2.2' de gizleme fonksiyonu olarak adlandırılan ve bilgi gizlemede en çok kullanılan yöntemler; "en önemsiz bite ekleme", "maskeleme ve filtreleme" ve "algoritmalar ve dönüşümler" olarak sıralanabilir [2].

2.4.4.1. En önemsiz bite ekeleme (LSB)

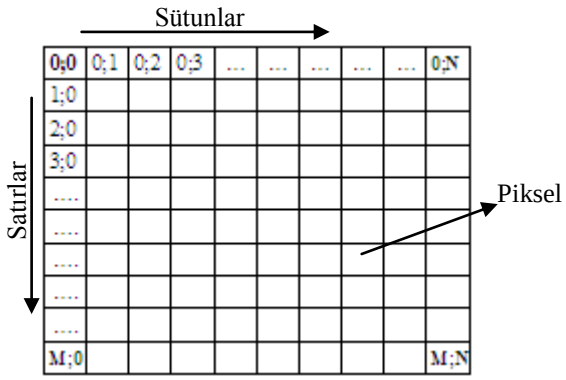
Veri gizleme tekniklerinin ilk uygulamalarından biri LSB tekniğidir [3]. En önemsiz bite ekleme yöntemi olarak adlandırılan LSB, yaygın olarak kullanılan ve uygulaması basit bir yöntemdir. Fakat yöntemin dikkatlice uygulanmaması durumunda veri kayıpları ortaya çıkmaktadır [14].

LSB yönteminde, resmi oluşturan her pikselin her baytının en önemsiz biti olan son biti, gizlenmek istenen verinin bitleri ile birer birer değiştirilmektedir. Burada her sekiz bitin en fazla bir biti değişikliğe uğratıldığından ve eğer değişiklik olmuşsa da değişiklik yapılan bitin byte'ın en az anlamlı biti olmasından dolayı, ortaya çıkan steganogramdaki (= örtü verisi + gömülü veri) değişimler insan tarafından algılanamaz boyutta olmaktadır [3,14].

Son bite ekleme işlemi resmin başından ya da sonundan olmak üzere sıralı bir şekilde yapılabileceği gibi, bir rasgele fonksiyon üreteci (random function generator) kullanılarak belirlenen bir piksel üzerinde değişiklik yapılması şeklinde de gerçekleştirilebilmektedir [14]. Yani veri ardışık olarak saklanabileceği gibi bir anahtar yardımıyla rastgele olarak seçilen piksellere de saklanabilir [9].

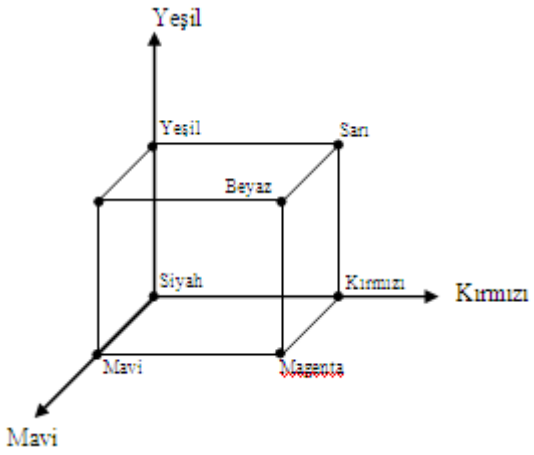
Resim dosyalarına bakıldığında; herhangi bir pikseli oluşturan bitlerden, en az önemli olan, en az anlam taşıyan bitler üzerinde yapılacak bir modifikasyon, insan duyularıyla algılanabilecek kadar etkili bir renk değişikliğine neden olmaz [3].

Sayısal resim, Şekil 2.3'te görüldüğü üzere, n satır ve m sütunluk bir dizi ile temsil edilir. Genellikle satır ve sütun indeksleri y ve x veya r ve c olarak gösterilebilir. 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 [8]. Gri seviyeli resimlerde ise her bir piksel 0 ile 255 arasında (8 bitlik) parlaklık seviyesi değerleri alır.



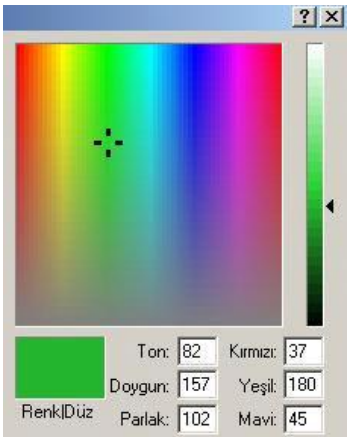
Şekil 2.3. Sayısal resmin temel yapısı

Sayısal resimler genel olarak 8, 16 veya 24 bitlik piksellerden oluşur. 24 bitlik piksellerden oluşan resimler en fazla veri saklayabilen resimlerdir. Bütün renk değerleri kırmızı, yeşil ve mavi (RGB) renklerin tonlarının karışımından elde edilir. Temel renklerin her biri bir bayt büyüklüğünde değerle ifade edilir ve 256' şar farklı renk tonuna sahiptir. Şekil 2.4' te renk küpü gösterilmektedir [9].



Sekil 2.4. Renk küpü

24 bitlik bir resmin her bir pikseli 3 ana rengi ifade eden 3 byte boyutundaki piksellerden oluşur. Şekil 2.5' te örnek bir rengi oluşturan kırmızı, yeşil ve mavi renk yoğunlukları gösterilmektedir. 1024x768 boyutundaki 24 bitlik bir resim dosyasında her bir byte değerinin en az önemli bitleri veri saklama icin kullanılırsa, toplam saklanabilecek veri boyutu, 1024x768x3/8 = 294912 bayt olacaktır [9].



Şekil 2.5. Bir rengi oluşturan kırmızı, yeşil ve mavi tonlara ait değerler

Resim içerisine LSB modifikasyonu ile veri saklamaya ilişkin bir örnek vermek gerekirse; saklanacak olan 9 bitlik bir verinin aşağıdaki bit değerlerinden oluştuğunu varsayalım.

1	1	0	1	0	0	1	1	0

Şekil 2.6. Lsb ile gizlenecek veriye ait bit değerleri

Taşıyıcı 24 bitlik bir BMP resmi olsun. Bu durumda saklanacak olan mesajı barındırmak için 3 piksel, yani 9 byte değerinin son bitleri yeterli olacaktır. Taşıyıcı dosyanın veri barındıracak olan piksellerinin en az önemli bitlerinin sırasıyla şu şekilde olduğunu varsayalım.

	1.Piksel			2.Piksel			3.Piksel		
	R	G	B	R	G	B	R	G	B
en az önemli bit	0	1	0	0	1	1	0	1	1

Şekil 2.7. Taşıyıcı resimde gizleme için kullanılacak en az önemli bit değerleri

LSB modifikasyonundan sonra bu piksellerin en az önemli bitlerinin değerleri şu şekilde olacaktır.

	1.Piksel			2.Piksel			3.Piksel		
	R	G	B	R	G	B	R	G	B
en az önemli bit	1	1	0	1	0	0	1	1	0

Şekil 2.8. Lsb işlemi sonucu taşıyıcı piksellerde oluşan bit değişimleri

Görüldüğü gibi 9 bit değerinin 6 tanesi (kırmızı ile belirtilenler) değişmiştir, ancak bu tür değişimler görsel olarak ayırt edilebilir olmamaktadır. Geri getirme işleminde yeni piksel değerlerinin en az önemli bitleri sırayla okunup yan yana dizildiğinde saklanan mesaj doğru bir şekilde elde edilecektir.

2.4.4.2. Maskeleme ve filtreleme

Maskeleme ve filtreleme tekniklerinde resmin, görüntü işleme teknikleri kullanılarak veri saklamaya en uygun alanları belirlenir ve saklama işlemi bu bölgelerde gerçekleştirilir. Genellikle 24 bitlik gri seviyeli resimlerde başarılı olan bir yöntemdir [15].

Maskeleme ve filtreleme teknikleri genellikle 24 bit ve gri-seviye görüntüler üzerinde işaretleme ve filigran yapılarak uygulanmaktadır. Teknik olarak filigran bir steganografik biçim değildir.

Maskeleme işlemi, sıkıştırma, kırpma ve bazı görüntü işlemleri açısından son bite ekleme yönteminden daha etkilidir. Maskeleme teknikleri belirli alanlara bilgileri gömer. Maskeleme tekniklerinin JPEG görüntülerde kullanılması daha uygundur.

2.4.4.3. Algoritmalar ve dönüşümler

Son bite ekleme yöntemi bilgi gizlemek için oldukça kolay ve hızlı bir yöntemdir, fakat görüntüye uygulanan işlemler ya da kayıplı sıkıştırmalar sonucunda bilgi zarar görebilmektedir. Dönüşümler ise yine daha çok JPEG dosyalar üzerinde kullanılmaktadır. En yaygın olarak kullanılan dönüşümler ise DCT(Ayrık Kosinüs Dönüşümü) ve DFT(Ayrık Fourier Dönüşümü)'dir [16].

Yüksek kalitedeki resimlerin sıkıştırılarak, örneğin JPEG formatı kullanılarak internet üzerinden gönderilmesi daha uygundur. Bunun için gizlenen bilginin kaybolmaması ve görüntünün sıkıştırılmasını sağlayan bazı yöntemler ve steganografik araçlar ortaya çıkarılmıştır. Gizlenecek veri/mesaj imgeler başka bir uzaya dönüştürüldükten sonra imgenin belirli alanlarına gömülmektedir. Dönüşüm tekniklerinde sıklıkla kullanılan yöntemlerin başında Ayrık Kosinüs Dönüşümü (Discrete Cosinus Transform-DCT), Ayrık Fourier Dönüşümü (Discrete Fourier Transform-DFT) ve Ayrık Dalgacık Dönüşümü (Discrete Wavelet Transform-DWT) gelmektedir [17].

Ayrık kosinüs dönüşümü ve ayrık fourier dönüşümü uygulamalarında imgeler 8x8'lik bloklara ayrılır ve her bir bloğa ayrık kosinüs dönüşümü veya ayrık fourier dönüşümü uygulanır. Dönüşüm sonucunda her bir blok için sol üstten sağ alt köşeye zig-zag'lar çizen yüksek frekanslı katsayılardan alçak frekanslı katsayılara giden katsayılar elde edilmektedir. Yüksek frekanslı katsayılar imgenin ortalama bilgilerini saklamaktadır. Nicemlendirme sonucunda küçük sayılar elde edilebilmektedir. Bu katsayılarda yapılacak değişiklik bloğun tamamını etkilediğinden bu katsayılara veri gizleme işlemi yapılmamaktadır. Alçak frekanslı katsayılar genellikle sıfır değerini aldığından, gizlenecek verinin kaybolması söz konusu olacaktır. Bu nedenle bu alanda veri gizlemesi yapılamamaktadır. Ancak orta frekanslarda verinin gizlenmesi yapılabilir. Ayrık kosinüs dönüşümü uygulamaları, en önemsiz bite ekleme yöntemi ile birleştirilerek de kullanılabilmektedir. Bu şekilde LSB yönteminde meydana gelen hata oranlarının azaltılması sağlanır.

Ayrık dalgacık dönüşümü uygulamalarında ise imge yüksek ve alçak geçirgen filtrelerden geçirilerek 4 parçaya ayrılır. Bunlar; imgeye ait özelliklerinin ortalamasının, dikey bilgilerinin, yatay bilgilerinin ve köşegensel bilgilerinin tutulduğu parçalardır. Gizlenecek veri bu parçalardan imgenin görsel olarak en az etkilendiği dikey, yatay ve köşegensel gibi alçak frekanslı bantlardaki katsayılara gömülmektedir.

Renkli resimler içerisine veri gizleme işlemi, imgelerin ayrık dalgacık dönüşümü katsayılarının eşik seviyesinden geçirilmesi ve DWT katsayılarının bloklara

ayrılmasıyla gerçekleştirilebilmektedir. Renkli imegelerin içine gizli veri saklanmasında diğer bir yöntem, örtü verisi ve gizlenecek mesajın ikisine de ayrık dalgacık dönüşümü uygulanması ve gizli mesaj dalgacık katsayıları, örtü verisi dalgacık katsayılarının orta frekans bandına gizlenmesidir.

DWT yöntemlerinde, imgelerin bloklara bölünmesinden ve yüksek ile alçak frekans bantlarında veri gizlenememesinden kaynaklanan nedenlerden dolayı düşük oranlarda veri gizlenebilmektedir. Bu kısıtlamanın en aza indirgenmesi maksadıyla çalışmalar yapılmıştır. Bu maksatla yapılan çalışmalardan bazıları imgenin özelliklerini en fazla içeren bölgelerden de faydalanmış, bazıları ise imgenin genel ve yerel özelliklerinden yararlanmıştır.

Dönüşüm tekniklerinin dezavantajı yüksek miktarlarda veri gömülememesidir. Buna karşın DCT, DFT, DWT gibi teknikler kırpma, sıkıştırma gibi bilinen ataklara LSB yöntemlerine göre daha dayanıklıdır.

2.4.5. Ses steganografi

İnsan duyma sistemi (İDS) aralığı yüzünden, ses sinyalleri içerisine bilgi gizleme oldukça uğraş gerektiren bir konudur. İDS 1/1.000'den daha büyük frekans aralığını fark edebilir. Aynı zamanda İDS nereden geldiği belli olmayan gürültülere de oldukça duyarlıdır.

Ses sinyalleri üzerinde uğraşırken ses dosyalarının hangi karakteristiklere sahip olduklarını bilmemiz gerekmektedir. Basit nicelendirme metodu ve geçici seçme oranı ses dosyalarının iki ana özelliğidir.

Basit nicelendirme metodu: Yüksek kaliteli sayısal seslerin 16-bit doğrusal quantisation ile ifadesinde en çok kullanılan yöntemdir. Bazı sinyal bozulmaları bu formatta ortaya çıkabilir [16].

Geçici seçme oranı: Ses için en çok kullanılan oranlar 8 kHz, 9.6 kHz, 10 kHz, 12 kHz, 16 kHz, 22.05 kHz ve 44.1 kHz ‘dir. Bu değer frekans aralığının kullanılabilecek en üst seviyesidir [16].

Ses dosyalarında veri gizleme yöntemleri ise şunlardır:

- Düşük bit kodlaması (Low-bit encoding)
- Aşama kodlaması (Phase coding)
- Taft yayılması (Spread spectrum)
- Yankı veri gizlemesi (Echo data hiding)

2.4.5.1. Düşük bit kodlaması

Görüntü steganografide kullanılan en önemsiz bite eklem yöntemiyle aynı şekilde gerçekleştirilir. Ses dosyasındaki verinin her baytının son bitine gizlenecek bilginin bir biti yazılır. Sonuçta oluşan değişiklik ses dosyasında gürültüye neden olmaktadır. Ayrıca dayanıksız bir yapısı vardır. Tekrar örnekleme veya kanalda oluşabilecek gürültü ile mesaj zarar görebilir [16].

2.4.5.2. Aşama kodlaması

Aşama kodlaması yöntemi resim dosyalarında uygulanan JPEG algoritması benzeri bir yapı taşımaktadır. Gömme işleminde ses dosyası küçük segmentlere bölünür ve her segmente ait aşama (faz) gizlenecek veriye ait aşama referansı ile değiştirilir.

Aşama kodlamasında, ses verisi n adet kısa segmente bölünür. Her segmente ayrık fourier dönüşümü (DFT) uygulanarak, aşama ve büyüklük (magnitude) matrisleri oluşturulur. Komşu segmentler arasındaki aşama farklılıkları hesaplanır. Her segment için yeni bir aşama değeri bilgi gizlenerek oluşturulur. Yeni aşama matrisleri ile büyüklük matrisleri birleştirilerek yeni segmentler elde edilir. Yeni segmentler birleştirilerek kodlanmış çıkış elde edilir [16].

2.4.5.3. Tayf yayılması

Ses sinyalinin kullandığı frekans tayfı üzerinde gizleme işlemini yapman bir tekniktir. Güçlü bir yapısı olamamakla birlikte seste gürültü meydana getirmektedir [16].

2.4.5.4. Yankı veri gizlemesi

Bilginin gizlenmesi taşıyıcı ses sinyali üzerine bir yankı eklenmesi ile sağlanmaktadır. Bilgi yankının gecikme miktarı, zayıflama oranı veya büyüklüğü gibi değerler kullanılarak gizlenir. İki farklı gecikme değeri kullanılarak insan kulağının algılamayacağı düzeyde 0 veya 1' in kodlanması mümkündür. Her bitin kodlanması için sinyal segmentlere bölünür. Yankı veri gizlemesi yöntemi herhangi bir gürültüye neden olmamakta veya kayıplı bir kodlama kullanmamaktadır.

BÖLÜM 3. KISMİ OPTİMİZASYON TEKNİĞİ İLE VERİ GİZLEME

3.1. Giriş

Steganografi, veri gizleme tekniklerinin bilgi güvenliği konusunda sunduğu yeni yaklaşımlarla son yıllarda önemini giderek artırmıştır. Steganografi yöntemlerinin tamamı, yazılım ile desteklenmekte ve güçlü algoritmalar ile ortaya çıkarılmaktadır. Bu açıdan, yeni yöntemler ile özellikle resim içerisine bilgi gömme teknikleri üzerine yoğunlaşılarak, bu alanda yeni yaklaşımlar elde edilmiştir. Kısmi optimizasyon yöntemi ile veri gömme tekniği de bu yeni yaklaşımlar arasındadır.

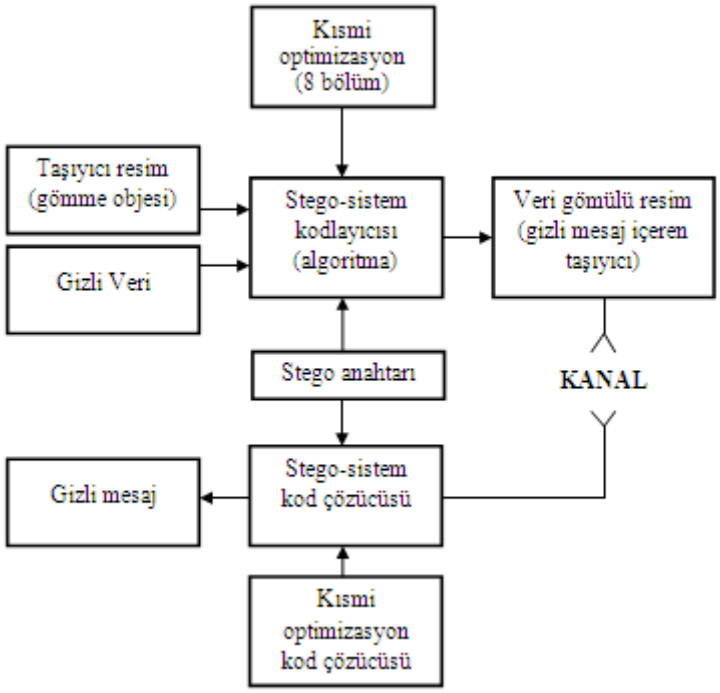
Akar'ın geliştirdiği kısmi optimizasyon yöntemi ile veri gömme tekniğinde taşıyıcı sekiz farklı optimizasyon bölgesine ayrılır ve her bir bölge sekiz farklı optimizasyon işlemine tabi tutulur. Tüm bölgeler için maksimum optimizasyon 2^8'e kadar artırılabilmektedir. Her bir bölgeden elde edilen sonuçlar en efektif değerlerdir ve bu nedenle değiştirilen (hata) bitlerin en aza indirgendiği sonucuna ulaşılmaktadır [4].

Kısmi optimizasyon tekniği yerine, klasik LSB kod çözme tekniği kullanıldığında elde edilen veri orijinal veri olmayacaktır. Bu durumda her bir bölgeye tek bir metot uygulanacağından bölgelerarası farklı kodlama teknikleri sebebiyle hatalı kod çözme işlemi gerçekleştirilebilir. Dolayısıyla iki farklı sonuca ulaşılmaktadır; birincisi minimum hata oranı için korunan orijinal piksellerin sayısının artırılması, ikincisi ise kendi içerisinde güvenliğin oluşturulmasıdır. Optimizasyon kodlaması ile bir çeşit bölgesel kodlama yapıldığından, her bir bölgede dinamik olarak gömme sırası değişeceğinden 256 farklı kodlama türü oluşmaktadır. Ancak kodlama için harcanan zaman, kodlama sayısı ile doğru orantılı olarak artacaktır. Bu nedenle bu uygulamada 2^3 yani 8 kodlama türü kullanılmıştır [4].

3.2. Kısmi Optimizasyon Tekniğinin İşlem Süreçleri

Şekil 3.1' de görüldüğü gibi kısmi optimizasyonun kodlama işlemi sürecinde beş bileşen bulunmaktadır [4]. Teknik açıdan maske olarak adlandırılan bir taşıyıcı bulunmakta olup "c" harfi ile gösterilmektedir. Gizlenmek istenen mesaj "m" harfi, kısmi optimizasyon ise "p" harfi ile sembolize edilmektedir. Bu bölümde maske olarak kullanılan resim sekiz farklı bölgeye ayrılmaktadır. Elde edilen gizli bilgi/mesaj ("m") içeren resim dosyası "stego-image" olarak adlandırılmakta ve "s" harfi ile gösterilmektedir. Son olarak stego şifreleme anahtarı "k" harfi ile sembolize edilmektedir. Veri gizleme sonucunda elde edilen çıkış "s"; $c + m + k + p$ stego-sistem kodlaması sonucunda oluşturulmaktadır [4].

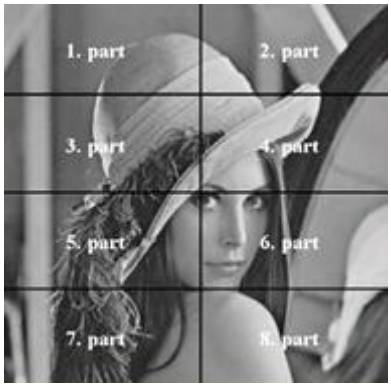
Kod çözme işlemi ise üç bileşenden oluşmaktadır. Kodlama işleminde kullanılan algoritma anahtarı bu durumda kod çözme sürecinde gereklidir.



Şekil 3.1. Kısmi optimizasyon stenografik sisteme ait veri gömme/çıkarma kodlayıcı ve kod çözücü blok diyagramı (Akar 2005)

Stego-sistem kod çözücüsü bölgesel optimizasyon kod çözücüsü tarafından kontrol edilmektedir. Bu bölümdeki kod çözme işlemi tamamlandıktan sonra kodlayıcıda kullanılan aynı anahtar ile kod çözme işlemine geçilmektedir. Her iki kod

çözümünde de doğru sonucu bulması ile orijinal mesaj çözülebilmektedir. Aksi halde orijinal mesaja ulaşılamaz [4].



Şekil 3.2. Sekiz farklı optimizasyon bölgesine ayrılmış olan gri seviyeli Lena resmi

Şekil 3.2' de veri gömme süreci başlangıcında sekiz farklı optimizasyon bölgesine ayrılmış olan gri seviyeli Lena resmi görülmektedir. Her bir bölge de kendi içerisinde yine sekiz farklı optimizasyona tabi tutulur. Örneğin kodlama işleminde birinci bölge Şekil 3.3' te görüldüğü gibi sekiz farklı optimizasyona tabi tutulmaktadır. Ancak kodlama için harcanan zaman bununla doğru orantılıdır. Bu nedenle bu uygulamada 2^3 yani 8 olarak seçilmiştir [4].

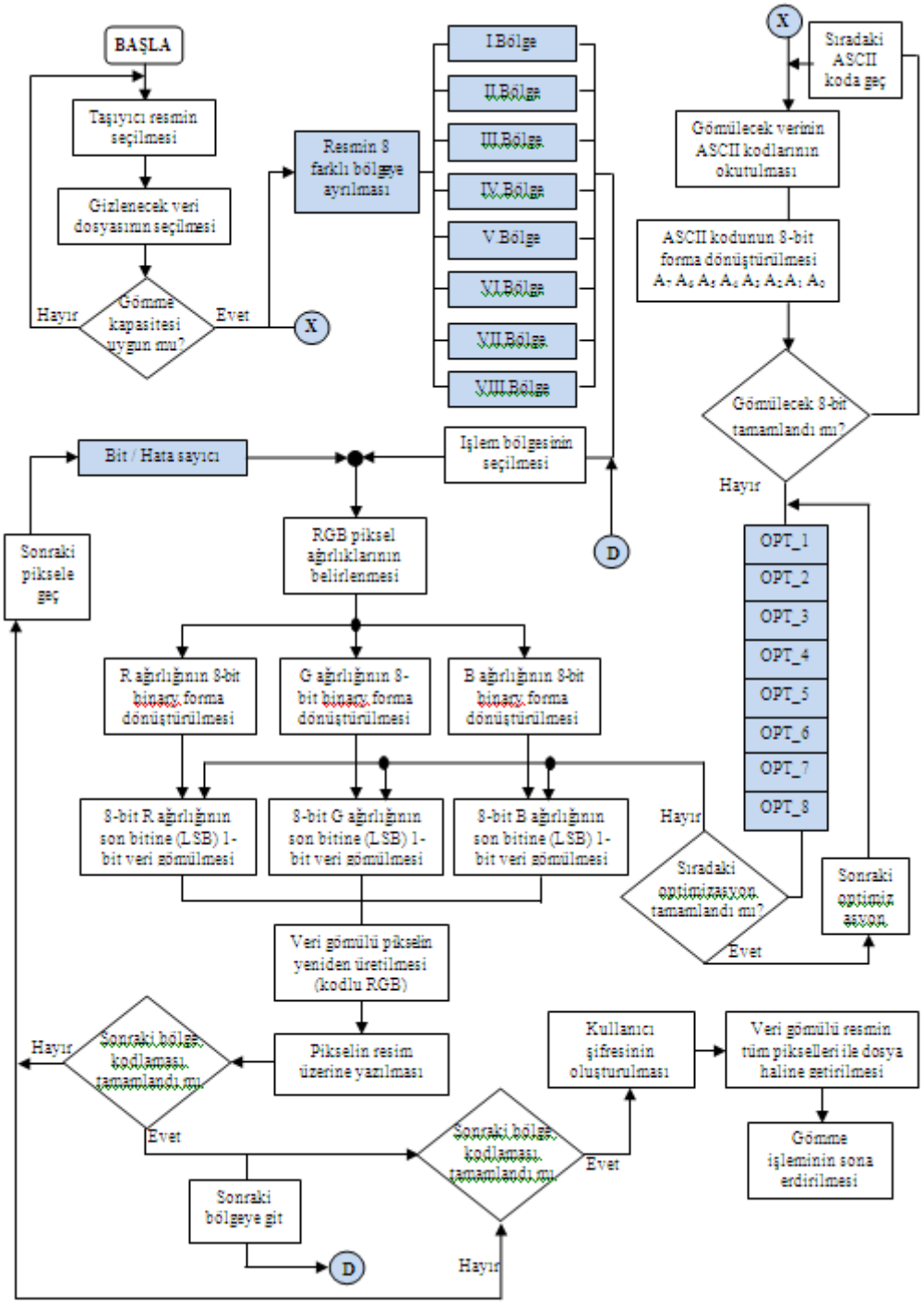
Optimizasyon 1:	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Optimizasyon 2:	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
Optimizasyon 3:	D_7	D_6	D_5	D_4	D_0	D_1	D_2	D_3
Optimizasyon 4:	D_0	D_1	D_2	D_3	D_7	D_6	D_5	D_4
Optimizasyon 5:	D_7	D_5	D_3	D_1	D_6	D_4	D_2	D_0
Optimizasyon 6:	D_0	D_2	D_4	D_6	D_1	D_3	D_5	D_7
Optimizasyon 7:	D_1	D_3	D_5	D_7	D_0	D_2	D_4	D_6
Optimizasyon 8:	D_6	D_4	D_2	D_0	D_7	D_5	D_3	D_1

Şekil 3.3. D_0–D_7 arasında bir ASCII kodunun gömülmesi önerilen bit düzenleri

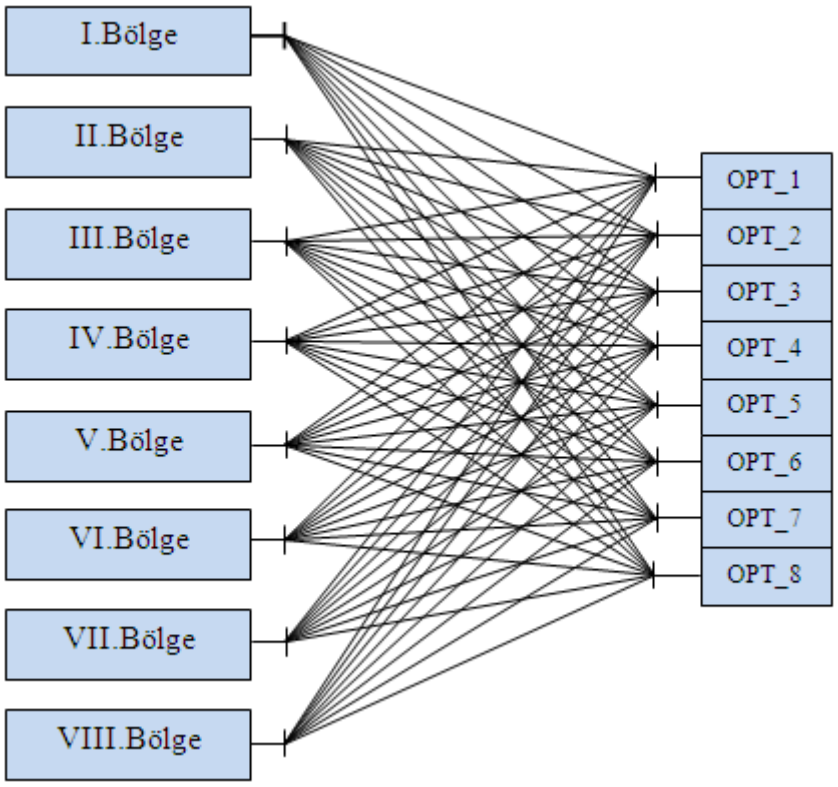
Şekil 3.3'te görüldüğü gibi, birinci optimizasyon için "D7, D6, D5, D4, D3, D2, D1, D0" sırası seçilmiş olup, ikinci optimizasyonda bu sıra "D0, D1, D2, D3, D4, D5, D6, D7" olarak değiştirilmektedir.

3.3. Kısmi Optimizasyon Yöntemi ile Veri Gömülmesi Katmanının Algoritması

Şekil 3.4'te kısmi optimizasyon tekniği ile ASCII kodun gömülmesi katmanının algoritması yer almaktadır [4]. Bu yapı incelendiğinde; uygulamanın başlatılması ile birlikte içerisine veri gömülecek olan bitmap (bmp.) uzantılı dosya belirlenir. Ardından gizli veri dosyası (doküman, ses, resim, sıkıştırılmış dosya vs.) seçilir. Gömen ve gömülen dosyalar arasında kapasite sorunu olup-olmadığı araştırılır. Sonuç olumlu ise 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 veri 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. 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 [18].



Şekil 3.4. Kısmi optimizaston yöntemi ile veri gömülmesi katmanının algoritması (Akar 2005)



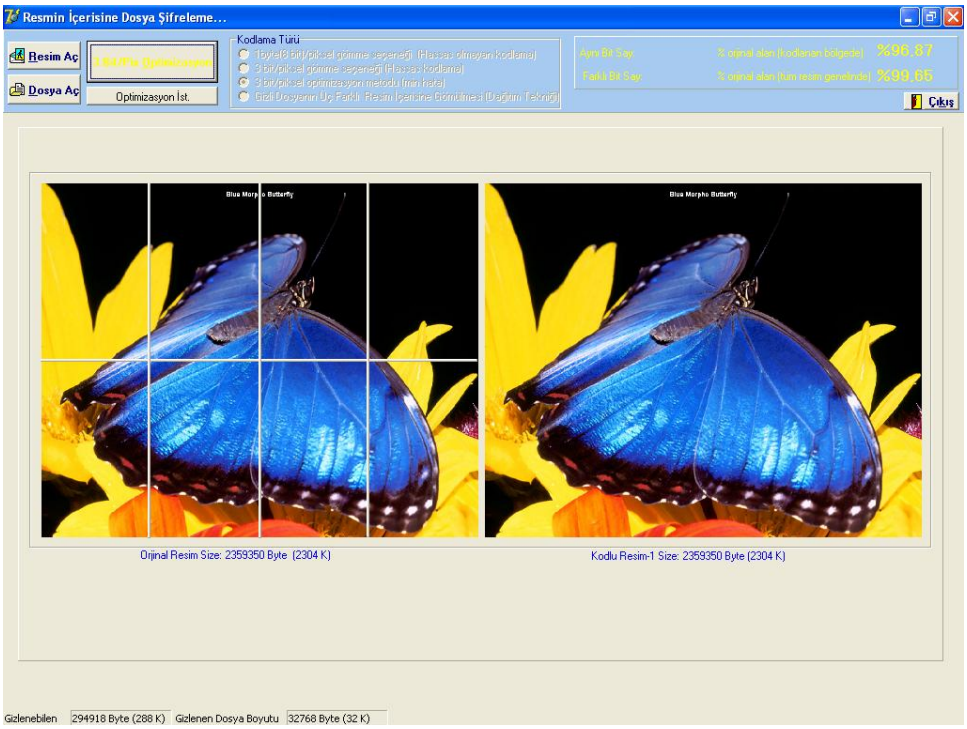
Şekil 3.5. Kısmi optimizaston yönteminde bölgeler ile optimizasyonlar arası geçişler (Akar 2005)

Şekil 3.4'te sunulan ve açıklanan algoritma katmanı kısmi optimizasyon yönteminde bölgeler ile optimizasyonlar arasındaki geçişlerin nasıl yapıldığı Şekil 3.5'te görülmektedir. Şekil dikkatlice incelendiğinde her bir bölgenin 8-farklı optimizasyon ile veri gömme işlemine tabi tutulduğu görülmektedir. Her bir bölgenin bu farklı veri gömme sıraları ile işlevsel olarak çalıştırılması neticesinde elde edilen bölgesel resmin orijinal resim ile karşılaştırılarak, hata bitlerinin saydırılması işlemi yapılmaktadır. Akar'a göre böylelikle tüm bölgelerin en verimli veri gömme yöntemini veren optimizasyon türü belirlendiğinden toplamda da en az hata bitine ulaşılması sağlanmaktadır.

Her bir bölgeden elde edilen sonuçlar en efektif değerler olup, değiştirilen (hata) bitlerinin en aza indirgendiği sonucunu yansıtmaktadır. Bu yöntemle sekiz bölgeden elde edilen değerler minimal değerler olup bit/hata oranı da minimum olmaktadır. Bu durum Tablo 3.1'de gösterilmektedir. Burada sekiz farklı bölgeye bu defa da sekiz farklı optimizasyondan hangi metot en efektif sonucu üretmiş ise bu bölge ve optimizasyon yöntemi resim içerisine gömülmektedir. Bu veriler resim içerisinden

alınarak bölgelere göre seçilen optimizasyon kriterleri değerlendirilmektedir. Dolayısıyla kod çözme işlemi sürecinde bu veriler referans olarak kullanılmaktadır. Bu yapılmayıp sıradan LSB kod çözme tekniği kullanıldığı takdirde elde edilecek veri, orijinal veri olmayacaktır. Çünkü bu durumda her bir bölgeye tek bir metot uygulanacağından bölgelerarası farklı kodlama teknikleri sebebiyle hatalı kod çözme işlemi gerçekleştirilecektir.

Tablo 3.1'de optimizasyon uygulamaları ve sonuçları görülmektedir. Burada OPT_1 olarak adlandırılan 1. sütunda klasik LSB tekniğinin de kullandığı sırayı içeren birinci optimizasyon verileri görülmektedir. Bu sütun dikkatlice incelendiğinde; bütün resmin bu yöntem ve kodlama sırasıyla kodlandığı varsayılırsa optimum sonuca ulaşılamayacağı açıktır. Bu örneklemede LSB klasik kodlama sütunu sekiz farklı optimizasyonun hiçbirisinde minimum hata oranını yakalayamamıştır. Buradan en az hata oranına sahip olamadığı anlaşılmaktadır. Tüm bölgelere LSB yaklaşımının uygulanması ile ne yazık ki minimum hata/piksel oranı elde edilmektedir. Yapılan optimizasyon tarama tekniği ile ilk bölge için OPT_7 en optimum sonucu vermiştir. Çünkü burada 13556 bit hatalı 19204 bit ise aynen korunarak toplam %58,62'lik orijinal olarak korunan bit yüzdesi elde edilmiştir. Benzer şekilde ikinci bölgede OPT_8 en optimum sonucu vererek % 54,25'lik bir başarı sağlamıştır. Bu örnek uygulamada tüm bölgelerdeki kazanılan hata önleme bitlerinin toplamı 554 olarak hesaplanmıştır. Bunun anlamı; içerisine veri gömülen resmin LSB tekniğinde en az 200 piksel, en çok ise 548 piksel daha fazla bozulmaya yol açtığı; ancak, optimizasyon metodu ile bu bitlerin orijinal olarak korunduğu Tablo 3.1'de görülmektedir [4].



Şekil 3.6. Bölgesel optimizayon yöntemi ile veri gömme uygulaması: örnekleme

Kısmi optimizasyon yöntemi ile veri gömme uygulamasında seçilen örnekleme Şekil 3.6'da görülmekte olup, elde edilen sonuçlar Tablo 3.1'de verilmiştir.

Tablo 3.1. Örnekleme için elde edilen optimizasyon sonuçları

Kısmi optimizasyon yöntemi ile elde edilen kazanımın tesadüfî olmadığı, farklı resimlerle gerçekleştirilen gömme uygulamaları üzerinde denenerek oluşan verilerle ispatlanmıştır [4].

Kısmi optimizasyon yöntemini kısaca özetlemek gerekirse, bu yöntemle resim 8 farklı bölgeye ayrılarak ilk önce birinci bölgeyi oluşturan piksellerin RGB ağırlıkları belirlenir. Eş zamanlı olarak gömülecek dosyanın ASCII kodları da belirlenmektedir. Bu bölgeye sekiz 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.

Bu teknikte herhangi bir sıkıştırma metodu uygulanmamaktadır. Gömülecek verinin en uygun sıkıştırma programı ile minimize edilerek geliştirilen program tarafından bölgesel optimizasyona tabi tutulması en efektif sonucu verecektir. En verimli sonucun alındığı bir sıkıştırma programı ile gömülecek dosya minimum büyüklüğe indirgenerek daha sonra geliştirilen yazılım ile bölgesel gömme işlemine tabi tutularak en iyi neticeye ulaşılabilir.

Bu tez çalışmasında, Akar'ın geliştirmiş olduğu kısmi optimizasyon tekniği ile veri gizleme uygulaması, sıkıştırma algoritması ile desteklenmiş, sıkıştırma algoritması tabanlı bir veri gizleme uygulaması gerçekleştirilmiştir. Dosya sıkıştırması yapmak için LZW sıkıştırma algoritması kullanılmış ve kısmi optimizasyon tekniği uygulamasında gizlenecek verinin önce boyutunun sıkıştırılarak minimize edilmesi sağlanmıştır. Böylece gizleme işleminin yapılmasıyla daha elverişli ve efektif bir sonuç elde edilmiştir.

BÖLÜM 4. VERİ SIKIŞTIRMA TEKNİKLERİ

Veri sıkıştırma, verilerin hafıza alanında daha az yer işgal etmeleri ve bir iletişim ağı üzerinden daha hızlı transfer edilebilmeleri için yaygın olarak kullanılmakta olan tekniklerdir. Bu sayede bellek üzerinde yer tasarrufu, veri aktarımında da zaman tasarrufu yapılabilmektedir. Veri sıkıştırma yöntemleri farklı temellere, farklı tipte verilere göre farklı sonuçlar üretse de temelde hepsi "gereksiz verileri yok etme" prensibine dayanmaktadır.

Son yıllarda disk kapasitelerinin hızlı bir şekilde artması, genel amaçlı sıkıştırma uygulamalarının kullanım oranını azalttıysa da, aslında sabit disklerimizde sakladığımız ses, görüntü ve hareketli görüntü dosyalarının tamamına yakını çeşitli yöntemlerle sıkıştırılmış haldedir [5].

Diskler ve teypler gibi saklama birimleri üzerinde saklanan, ya da bilgisayar haberleşme hatlarından iletilen veriler, önemli ölçüde artıklık içerirler. Veri sıkıştırma algoritmalarının amacı, bu artıklıkları kodlayarak bilgi kaybı olmaksızın, veri yoğunluğunu artırabilmektir. Dört tür artıklık mevcuttur [19].

4.1. Veri Artıklık Türleri

4.1.1. Karakter dağılımı

Herhangi bir karakter katarında, bazı karakterler diğerlerine göre daha sık kullanılırlar. Özellikle sekiz bitlik ASCII kodlarının kullanıldığı özel bir dosyada, karakterlerin 3/4 ü kullanılmayabilir. Sonuçta her bir sekiz bitlik paketin, iki bitinden tasarruf edilmiş olur. Bir envanter kaydında, sayısal değerler çok daha yaygındır (ikili ya da ondalık sayılar istatistiği değiştirebilir) ve kendilerine ayrılmış alandaki

sınırlamalar, dosyadan dosyaya, önemli ölçüde değişebilecek karakter dağılımına neden olabilir. Örneğin, ambar yerlerinin adreslenmesi için alfabetik veya sayısal değerlerin kullanılması, envanter dosyasındaki dağılımı değiştirebilir. Benzer şekilde, envanter kaydında yer alan açıklayıcı metinler, her karakter için gereken ortalama bit sayısını etkileyecektir [19].

4.1.2. Karakter tekrarı

Eğer bir karakter katarı, tek bir karakterin tekrarından oluşuyor ise veri, olduğundan daha yoğun bir şekilde kodlanabilir. Bu tür katarlar metin tipi dosyalarda fazla yer almamaktadır. Bununla beraber, formatlı iş dosyalarında kullanılmayan alanlar oldukça fazla miktarda yer almaktadır. Bir envanter kaydında, kısmen kullanılan alfabetik alanlardaki boşluk katarlarına, sayısal alanlarda sıfır katarlarına ve kullanılmayan alanlarda null katarlarına oldukça sık rastlanır. Grafik görüntüler, özellikle iş grafiklerindeki çizgiler çoğunlukla homojen boşluklardan oluşmuştur [19].

4.1.3. Çok kullanılan sözcükler

Belli karakter dizileri, diğerlerine göre daha fazla sıklıkta kullanılırlar ve bu nedenle olması gerekenden daha az bit sayısı ile temsil edilebilirler. Örneğin, ingilizcede ‘ze’ gibi pekçok karakter çiftinin oluşma olasılığı, tek harfin oluşma olasılığından fazladır ve daha az bit ile kodlanabilirler. Benzer şekilde, ‘gc’ gibi oluşma olasılığı az olan karakter çiftlerinin, daha uzun bit kombinasyonları ile kodlanması, bit kullanımını iyileştirecektir.

Bazı tip dosyalarda olduğu gibi (programlama dilleri kaynak programlarını içeren dosyalar, metin dosyaları, v.b.) belli anahtar kelimeler diğerlerinden fazla miktarda yer alır. Örneğin, bu tezde "sıkıştırma” ve “şifreleme” sözcükleri sıklıkla kullanılmaktadır. Envanter kayıtlarında, depo isimleri gibi belli alanlar tekrar tekrar kullanılır. Sayısal alanlar ise, sadece sayısal olan ve hiçbir harf, özel işaret içermeyen dizilerden oluşur. Eğer bir metin bu tip bir artıklık içeriyor ise, 8 bitlik kodların

kullanılması yerine, 4 veya daha az bitlik kodların kullanılması ile temsil edilebilir [19].

4.1.4. Konumsal artıklık

Eğer belli karakterler, her bir veri bloğu içinde, önceden tahmin edilebilir yerlerde bulunuyorsa, bunlar kısmi olarak artıklık taşımaktadır. Dikey bir çizgi içeren bir resimde, çizgi her taramada aynı yerde görüneceğinden, daha az bit dizileri ile kodlanabilir. Bir envanter dosyada belli kayıtlar, hemen hemen aynı değişkenlere sahiptir. Diğer taraftan metin tipi dosyalar durumsal artıklık içermezler.

Bu dört tip artıklık bazı durumlarda çakışmaktadır. Örneğin, içinde sıfırların fazla miktarda yer aldığı tamsayılardan oluşan bir kayıt, ilk üç tip artıklık türünü de içermektedir [19].

4.2. Sıkıştırma Biçimlerine Göre Veri Sıkıştırma Yöntemleri

Veri sıkıştırma yöntemleri, kayıplı ve kayıpsız sıkıştırma yöntemleri olmak üzere ikiye ayrılır.

4.2.1. Kayıplı sıkıştırma

Kayıplı sıkıştırmada, verinin bütünlüğünü en az düzeyde etkileyecek olan veri kümeleri çıkartılır, geriye kalan veri kümeleri kayıpsız sıkıştırmaya tâbi tutularak sıkıştırma sağlanır. Bir veri kayıplı bir sıkıştırma yöntemi ile sıkıştırılırsa, verinin tamamı değil, sadece belirli bir kısmı geri getirilebilir. Veri bire bir aynı şekilde geri getirilemediği için bu tür yöntemlere kayıplı yöntemler denir.

Kayıplı veri sıkıştırma genellikle belirli bir miktar veri kaybının insan gözü ve kulağı tarafından hissedilemeyeceği durumlarda, örneğin fotoğraf görüntüleri, video ve ses için kullanılır. İnsan gözü ve kulağı yüksek frekans değerlerine daha az hassasiyet gösterdiği için, genellikle veri eleme işlemi yüksek frekans değerlerinin simgeleyen veriler üzerinde yapılır.

4.2.2. Kayıpsız sıkıştırma

Kayıpsız sıkıştırma yöntemleri, orijinal veri ile sıkıştırıldıktan sonra geri getirilecek olan verinin tamamıyla aynı olmasının gerekli olduğu durumlarda kullanılır. Örneğin metin tipinde veriler kayıpsız olarak sıkıştırılmalıdırlar, çünkü geri getirildiklerinde kelimelerinde veya karakterlerinde eksiklikler olursa, metnin okunabilirliği azalacak ve hatta anlam kayıpları meydana gelebilecektir. Kısacası, insan gözünün ve kulağının hassasiyeti ile direkt olarak ilgisi bulunmayan, metin belgeleri, kaynak kodları, çalıştırılabilir program dosyaları gibi dosyalar kayıpsız sıkıştırılmak zorundadırlar.

İki tip kayıpsız sıkıştırma yöntemi vardır. Bunlardan ilki değişken uzunluklu kodlama olarak da bilinen olasılık (veya istatistik) tabanlı kodlama, ikincisi ise sözlük tabanlı kodlamadır.

4.3. Uygulama Alanlarına Göre Veri Sıkıştırma Yöntemleri

4.3.1. Metin ve ikili tabanlı veri sıkıştırma

Bu alanda veriler kayıpsız olarak sıkıştırılmalıdırlar, çünkü sıkıştırma açma işlemi uygulandığında kelimelerde veya karakterlerde eksiklikler söz konusu olacağından, metnin okunabilirliği azalacak ve metinde anlam kayıpları meydana gelebilecektir. Dolayısıyla her bit değerli olduğu için kayıpsız yaklaşımların kullanılması şarttır [20].

4.3.2. Ses verisi sıkıştırma

Sayısal olarak kaydedilmiş ses sinyallerinin kayıplı ya da kayıpsız olarak daha düşük boyutta kaydedilmesi işlemidir. İyi sıkıştırma oranları sağladığı için genellikle kayıplı yaklaşımlar tercih edilirken, her bitin önemli olduğu profesyonel amaçlı uygulamalar için ise kayıpsız yaklaşımlar kullanılır.

4.3.3. Görüntü verisi sıkıştırma

Görüntü sıkıştırma, görsel bir öğeyi depolamak için kullanılan ve görüntüyü elektronik olarak depolamak için gerekli sayısallaştırılmış bilgi miktarını azaltan bir tekniktir. Şekiller ve taranmış metin görüntüleri gibi düşük frekanslı görüntülerde kayıpsız, fotoğraflarda ise genellikle kayıplı sıkıştırma kullanılır.

4.3.4. Hareketli görüntü (video) sıkıştırma

Sıkıştırılmamış hali çok büyük olan video dosyaları, kayıpsız yöntemler ile düşük oranlarda sıkıştırılabildikleri için, hareketli görüntü sıkıştırma işlemi çoğunlukla kayıplı yöntemler ile gerçekleştirilir. Bununla birlikte hızlı kodlama ihtiyacı olduğu durumlarda veya en küçük bir kaybın bile istenmediği profesyonel amaçlı uygulamalar için kayıpsız sıkıştırma da gerekli olabilmektedir.

4.4. Olasılık Tabanlı Veri Sıkıştırma Teknikleri

Sıkıştırılması istenen mesajın (semboller kümesinin) tek tek tüm sembollerinin veya birkaç sembolün bir araya getirilmesi ile oluşturulan alt sembol kümelerinin olasılıklarının bulunması, ve bu olasılık dağılımlarını temel alarak mesajın tekrar kodlanmasına *olasılık kodlaması* ve buna dayalı tekniklere de *olasılık tabanlı teknikler* adı verilir [5].

Olasılık tabanlı kodlamada, sıkıştırılan verinin bütünü içinde daha sık kullanılan sembollere bit adedi olarak daha küçük boyutta kodlar atanması prensibi ile sıkıştırma yapılır [5]. En çok kullanılan olasılık tabanlı teknikler; Huffman Kodlaması ve Aritmetik Kodlama'dır.

4.4.1. Huffman kodlama

Huffman kodlaması, MIT'de Robert Fano tarafından kurulan sınıfta öğrenci olan David Huffman tarafından, verilen bir ödev üzerine, 1952 yılında geliştirilmiştir. GZIP, JPEG, MPEG gibi yaygın olarak kullanılan sıkıştırma yöntemlerinde son işlem olarak kullanılır ve sıkıştırma algoritmalarında en yaygın olarak kullanılan bileşendir. CCITT'nin faks iletimi için geliştirdiği 1-boyutlu kodlama, tipik bir Huffman kodlamasıdır [21].

Temelde bir metin içindeki karakterlerin frekanslarına göre sınıflandırılarak belirli bir kuralla oluşturulan bir ağaca göre atanmış bitlerle temsil edilmesine dayanır. Algoritma, en yüksek frekanslı karakterin en az bitle, en düşük frekanslının da en çok bitle temsiline dayanan bir yol izlemektedir.

Huffman algoritması, bir veri kümesinde daha çok rastlanan sembolü daha düşük uzunluktaki kodla, daha az rastlanan sembolleri daha yüksek uzunluktaki kodlarla temsil etme mantığı üzerine kurulmuştur. Veri kümesindeki sembol sayısına ve sembollerin tekrarlanma sıklıklarına bağlı olarak Huffman sıkıştırma algoritması %10 ile %90 arasında bir sıkıştırma oranı sağlayabilir.

Huffman tekniğinde semboller (karakterler) ASCII'de olduğu gibi sabit uzunluktaki kodlarla kodlanmazlar. Her bir sembol değişken sayıda uzunluktaki kod ile kodlanır. Bir veri kümesini Huffman tekniği ile sıkıştırabilmek için veri kümesinde bulunan her bir sembolün ne sıklıkta tekrarlandığını bilmemiz gerekir. Örneğin bir metin dosyasını sıkıştırıyorsak her bir karakterin metin içerisinde kaç adet geçtiğini bilmemiz gerekiyor. Her bir sembolün ne sıklıkta tekrarlandığını gösteren tablo frekans tablosu olarak adlandırılmaktadır. Dolayısıyla sıkıştırma işlemine geçmeden önce frekans tablosunu çıkarmamız gerekmektedir. Bu yönteme Statik Huffman tekniği de denilmektedir. Diğer bir teknik olan Dinamik Huffman tekniğinde sıkıştırma yapmak için frekans tablosuna önceden ihtiyaç duyulmaz. Frekans tablosu

her bir sembolle karşılaştıkça dinamik olarak oluşturulur. Dinamik Huffman tekniği daha çok haberleşme kanalları gibi hangi verinin geleceği önceden belli olmayan sistemlerde kullanılmaktadır.

4.4.2. Aritmetik kodlama

Shannon 1948'deki makalesinde, aritmetik kodlama teorisinin ispatına çok yakın bir kavramdan bahsetmiştir. Fano'nun MIT'deki bilişim teorisi sınıfının bir başka öğrencisi olan Peter Elias ise bu fikrin öz yinelemeli bir uyarlamasını gerçekleştirmiştir. Ama kendisi bu çalışmasını hiç yayınlamadığı için, biz bunu Abramson'un bilişim teorisi üzerine yayınladığı kitabından bilmekteyiz. Daha sonra Jelinek tarafından yazılan bir başka kitabın ekler kısmında ise, aritmetik kodlama fikri değişken uzunluklu kodlamanın bir örneği olarak yer almıştır. Sonlu duyarlık sorununun çözülmesi, modern aritmetik kodlamanın başlangıcı olmuştur. Pratik aritmetik kodlama algoritmaları ve veri sıkıştırmada kullanılması hakkında birçok makale yazılmıştır [21].

Aritmetik kodlamanın temel fikri, *n* adet mesajın her olası serisini temsil etmek için 0 ile 1 arasındaki bir sayı aralığını (örneğin 0.2 ile 0.5 aralığı gibi) kullanmaktır. Alfabenin küçük olduğu ve karakterlerin belirme olasılığında büyük farklar olduğu durumlarda, Huffman Kodlaması etkinliğini yitirirken, Aritmetik kodlama bu durumlarda daha başarılıdır.

Aritmetik kodlamada, belirli bir sembol serisini diğer sembol serilerinden ayırmak için, her serinin tekil bir belirleyici ile etiketlenmesi gerekir. Bu etiket, genellikle 0 ile 1 arasında bir sayı şeklinde belirlenir.

4.5. Sözlük Tabanlı Veri Sıkıştırma Teknikleri

Sözlük tabanlı kodlamada ise, sık tekrarlanan sembol grupları için tek bir sembol kullanılması ile sıkıştırma yapılır. Bir metinde sıkça tekrar eden kelimeler, bir

görüntü dosyasında tekrar eden piksel grupları gibi, yinelenen kalıpların belirlenmesi ve bu kalıplardan bir sözlük oluşturularak, her kalıbın sözlükteki sıra numarasının kodlanmasına dayalı tekniklerdir. En çok kullanılan sözlük tabanlı teknikler; LZ77, LZ78 ve LZW yöntemleridir. Sözlük tabanlı teknikleri, static, yarı-statik ve dinamik olmak üzere 3 temel kategoriye bölmek mümkündür.

Kaynak hakkında önceden önemli oranda bilgi varsa, sözlük oluşturma masrafından kurtulmak için statik sözlük kullanılabilir. Eğer kaynak hakkında bilgi yoksa, dinamik veya yarı-statik yaklaşımlardan birini kullanmak daha etkili olacaktır.

4.5.1. Statik sözlük yaklaşımı

Sıkıştırılacak her verinin aynı sözlük ile sıkıştırılması statik sözlük yaklaşımıdır. Daha çok ASCII standardında kodlanmış metin tipinde verilerin sıkıştırılmasında kullanılır. Örneğin noktadan sonra boşluk (._) veya virgülden sonra boşluk (,_) gibi her tip metinde sıkça geçen karakter grupları, "_ve_", "_veya_" gibi kelimeler, ASCII Tablosunda genellikle kullanılmayan karakterlerin yerine yerleştirilebilir.

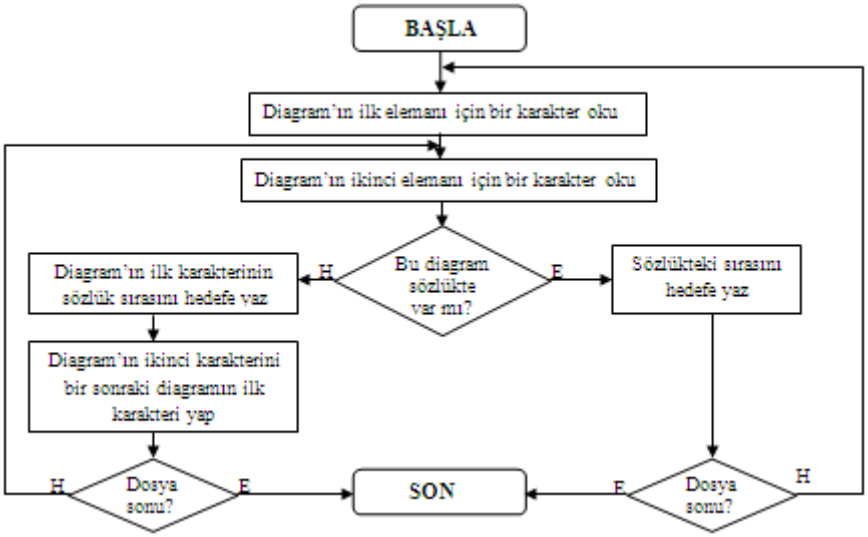
Statik sözlük hem sıkıştırma hem de açma algoritmalarında sabit olarak bulunacağı için, önceden bir veya daha fazla sayıda geçiş yaparak sözlük oluşturma işlemine gerek yoktur. Statik yaklaşımlar sıkıştırılacak veriye göre uyarlanabilir bir yapıda olmadıkları için dinamik yaklaşımlar kadar yüksek oranda sıkıştırma yapamasalar da, onlara göre çok daha hızlı çalışırlar. En çok bilinen static sözlük yaklaşımı diagram kodlamasıdır [5].

4.5.1.1. Diagram kodlaması

Diagram Kodlaması belirli bir kaynak tipine bağımlı olmayan bir statik sözlük tekniğidir. Bu kodlamada, kaynakta bulunabilecek tüm harflerle ve en sık kullanılan ikili karakter grupları (digram) ile oluşturulan bir statik sözlük kullanılır. Örneğin,

kaynakta kullanılan dilin alfabesindeki tüm büyük harfler, küçük harfler, rakamlar ve noktalama işaretleri sözlüğün ilk kısmına ve istatistiksel bir analiz sonucunda bulunabilecek olan bu dildeki en sık tekrar edilen ikili karakter grupları sözlüğün ikinci kısmına yerleştirilebilir. Eğer ikili değil de üçlü karakter grupları ile sözlük oluşturulduysa trigram kodlaması, n'li karakter grupları ile oluşturulduysa n-gram kodlaması olarak adlandırılır [5].

Şekil 4.1'de akış şeması verilen digram kodlayıcısı şu şekilde çalışır; kaynak dosyadan iki karakter okur ve bu karakterlerden bir diagram oluşturur. Bu diagramı sözlükte arar. Bulursa, sözlükteki sırasını hedef dosyaya yazar ve yeni bir diagram oluşturmak için iki karakter daha okur. Bulamazsa, ilk karakterinin sözlükteki sırasını hedef dosyaya yazar ve ikinci karakteri bir sonra aranacak diagramın ilk karakteri yapar. Dosyadan bir karakter daha okuyarak diagramı tamamlar. Döngü dosya sonuna kadar bu şekilde devam eder.



Şekil 4.1. Digram kodlayıcısının akış şeması (Mesut 2006)

Statik sözlük yaklaşımında, her metinde sıkça geçen karakter grupları veya kelimeler ASCII Tablosunda genellikle kullanılmayan karakterlerin yerine yerleştirilmektedir. Fakat bu yaklaşım, kullanılmadığını düşündüğümüz karakter eğer metinde kullanıldıysa hataya yol açacaktır. Bundan kaçınmak için;

- Sözlük büyüklüğü 256 karakter uzunluğundaki ASCII Tablosu ile sınırlı bırakılmayarak, 512 ya da 1024 karaktere kadar genişletilebilir.
- Hangi karakterlerin kullanılıp hangilerinin kullanılmadığı, sıkıştırılacak metin önceden bir defa okunarak tespit edilebilir (yarı-statik yaklaşım).

Bu işlemler bir miktar zaman kaybına neden olsa da, sıkıştırmanın performansına yapacağı olumlu etki nedeniyle, bu zaman kaybı göz ardı edilebilir.

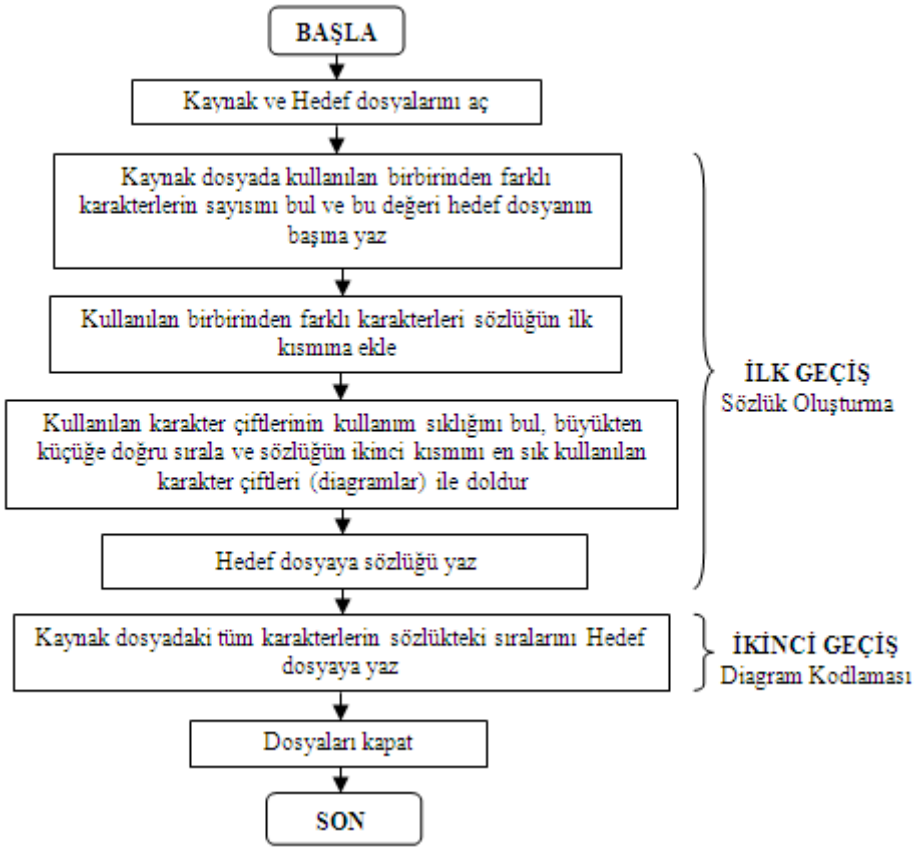
4.5.2. Yarı-statik sözlük yaklaşımı

Tek geçişli olan statik sözlük modelinde sıkıştırılacak olan tüm veriler aynı sözlük kullanılarak sıkıştırılırken, çift geçişli olan yarı-statik modelde, ilk geçişte sıkıştırılacak veride yer alan sembollerin dağılımları öğrenilir ve bu dağılama en uygun sözlük oluşturulur, ikinci geçişte ise bu sözlük kullanılarak sıkıştırma yapılır.

SSDC(Semi-Static Diagram Coding) ve ISSDC(Iterative Semi-Static Digram Coding) kodlamaları, yarı-statik sözlük yaklaşımına göre çaışırlar.

4.5.2.1. SSDC (Semi-static digram coding / Yarı-statik diagram kodlaması)

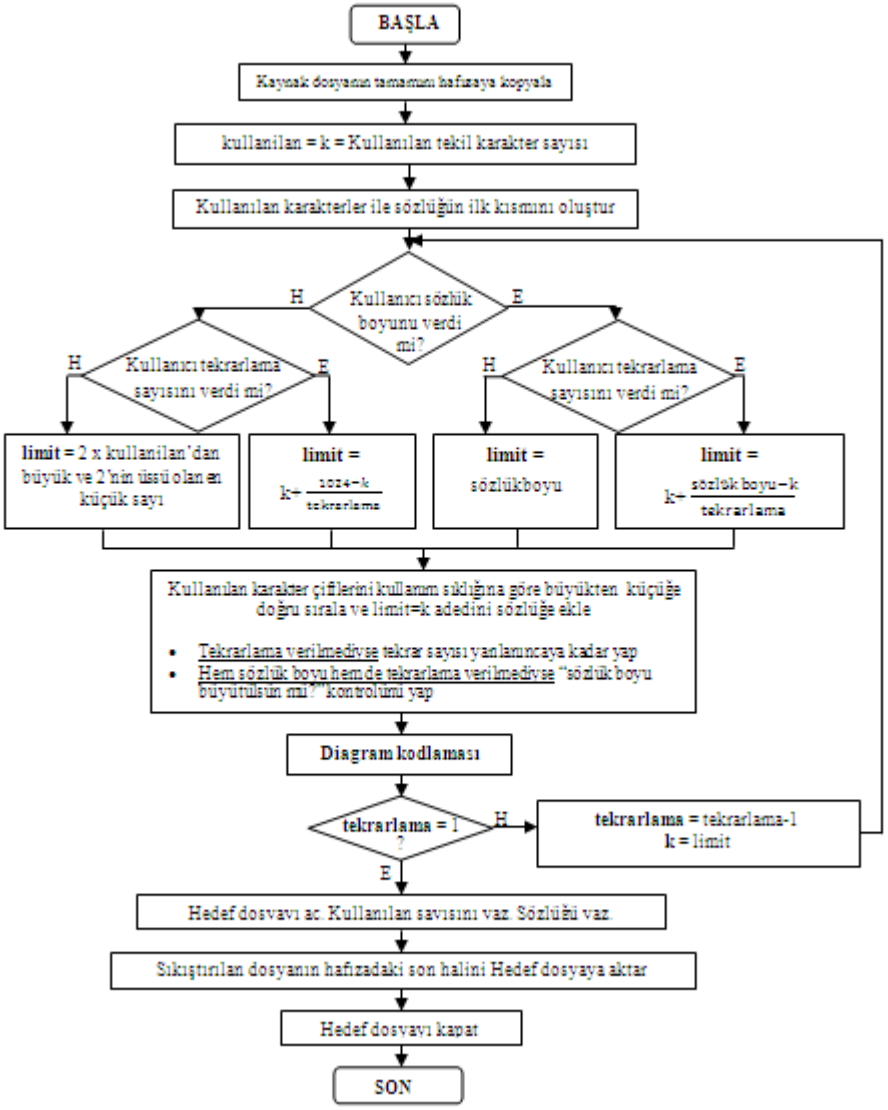
SSDC, digram kodlamasını temel alan, çift geçişli yarı-statik bir algoritmadır. Bu algoritma sözlüğü oluşturmak amacıyla bir ön geçiş yaptığı için, sıkıştırma zamanı tek geçişli statik sözlük yaklaşımına göre daha fazla olacaktır. Fakat kaynağa özel bir sözlük oluşturulduğu için, sıkıştırma oranı statik sözlük yaklaşımına göre daha iyi hale gelecektir [5]. SSDC sıkıştırma algoritmasının akış şeması Şekil 4.2'de verilmiştir.



Şekil 4.2. SSDC sıkıştırma algoritmasının akış şeması (Mesut 2006)

4.5.2.2. ISSDC (Iterative semi-static digram coding / Tekrarlı yarı-statik diagram kodlaması)

SSDC tabanlı çalışan ISSDC'de, tekrarlanan bir yaklaşım kullanılarak sıkıştırma oranı arttırılmıştır. Bu çok geçişli algoritma, sözlüğün ikinci kısmı olan digram kısmını tek bir geçişte değil, kullanıcı tarafından da belirlenebilen tekrarlama sayısı kadar geçişte doldurur. Her tekrarlama, kendisinden önce gelen tekrarlamaların digram'lar ile doldurduğu sözlük kısımlarını da sözlüğün ilk kısmı gibi, yani tek karakterlik elemanlar gibi görür ve buna göre digram kodlaması kullanarak sıkıştırma yapar. Böylece, $2^{(\text{tekrarlama sayısı})}$ karakterlik sık tekrar edilen bir karakter grubu bile, hedef dosyada tek bir karakter ile temsil edilebilir [5].



Şekil 4.3. ISSDC sıkıştırma algoritmasının akış şeması (Mesut 2006)

Eğer tekrarlama sayısı ve sözlük büyüklüğü verildiyse, otomatik olarak karar verilmesi gereken bir durum yoktur. Bu durumda, kaynakta kullanılan karakterler sözlüğe eklendikten sonra, sözlükte kalan boş yer tekrarlama sayısına bölünerek, yapılacak tekrarlama adımında sözlüğün ne kadar büyüklükte bir kısmının doldurulacağı belirlenir. Bu değer, kullanılan karakter sayısına (k) eklenerek bir limit değeri bulunur ve sözlüğün [k, limit] aralığı kaynakta en çok tekrarlanmış olan karakter çiftleri ile doldurulur. Karakter çiftleri, yani digram'lar, kullanım sayılarına göre büyükten küçüğe doğru sıralı şekilde sözlüğe eklenirler (Şekil 4.3).

4.5.3. Dinamik (uyarlanır) sözlük yaklaşımı

Dinamik sözlük modelinde, tek bir geçişte hem sözlük oluşturulur, hem de sıkıştırma yapılır [5]. Dinamik sözlük yaklaşımı, statik sözlük yaklaşımı ve yarı-statik sözlük yaklaşımının avantajlı yönlerini bir araya getirmiştir. Statik sözlük yaklaşımı gibi tek geçişlidir dolayısıyla hızlıdır ve yarı-statik sözlük yaklaşımı gibi kaynağa özel sözlük üretir dolayısıyla sıkıştırma oranı yüksektir [22].

Dinamik sözlük tekniklerinin çoğu Jacob Ziv ve Abraham Lempel tarafından 1977 ve 1978 yıllarında yazılmış olan iki farklı makale üzerine geliştirilmişlerdir. 1977'deki makaleyi temel alan yaklaşımlara LZ77 ailesi, 1978'deki makaleyi temel alan yaklaşımlara ise LZ78 ailesi denir [23]. LZ78 ailesinin en çok bilinen ve en iyi sıkıştırma oranı sağlayan üyesi 1984 yılında Terry Welch tarafından yayınlanan LZW algoritmasıdır [5].

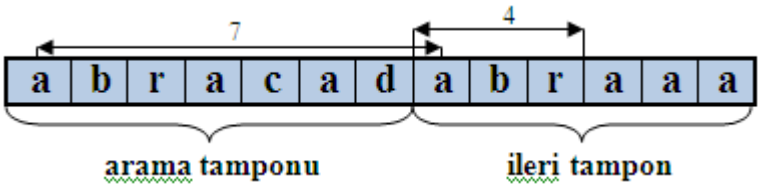
4.5.3.1. LZ77 sıkıştırma algoritması

Abraham Lempel ve Jakob Ziv tarafından geliştirilen ve 1977 yılında yayınladıkları "A Universal Algorithm for Data Compression" isimli makalelerinde tanımladıkları bu yöntem, o yıllarda tüm dünyada büyük ilgi görmüştür. Algoritmanın eksik yönleri zaman içinde farklı bilim adamları tarafından geliştirilmiştir. Sonraları yeni geliştirilen algoritmaların hepsine LZ77 ya da LZ1 ailesi denilmiştir.

LZ77 ailesi metin tabanlı veri sıkıştırmada büyük aşama kaydedilmesinin yolunu açmış, 80'li ve 90'lı yılların popüler sıkıştırma paketleri değişken uzunluklu kodlayıcı ile desteklenen LZ77 tabanlı algoritmalar kullanmışlardır [24].

LZ77 yaklaşımında sözlük, daha önce kodlanmış serinin bir parçasıdır. Algoritmadaki arama tamponunun büyüklüğü, daha önce kodlanmış serinin ne büyüklükte bir parçasında arama yapılacağını belirler. Arama tamponu büyütüldükçe, sıkıştırma oranı artar, fakat aynı zamanda sıkıştırma zamanı da artar.

Örneğin, "abracadabra" kelimesini LZ77 algoritması ile sıkıştıralım.



Şekil 4.4. LZ77'de arama tamponu ve ileri tampon

İleri tamponun ilk karakteri olan *a*, arama tamponunda sondan başa doğru aranır. İkinci karşılaştırmada benzerlik bulunur, fakat bu karakterden sonra *b* karakteri değil de *d* karakteri yer aldığı için benzerlik uzunluğu sadece 1'dir. Arama devam ettirilir. İki karakter sonra bir *a* daha bulunur, sonrasında *c* yer aldığı için bunun da benzerlik uzunluğu 1'dir. Aramaya devam edilir. Arama tamponunun başında, yani ileri tamponda aranan karakterden 7 uzaklıkta (offset=7) bir *a* daha bulunur. Bu defa benzerlik uzunluğu 4'tür (abra). İleri tamponda "abra" serisinden sonra yer alan *a* karakteri ile birlikte [7,4,C(a)] şeklinde üçlü olarak kodlanır. İleri tamponun en sonundaki *a* karakteri ise [0,0,C(a)] şeklinde kodlanır.

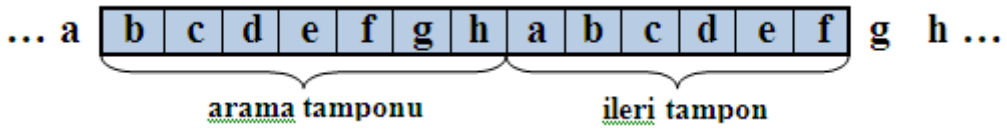
Sıkıştırma açma aşaması kodlanan tüm üçlü karakter gruplarının benzer şekilde açılması ile gerçekleştirilir. Mesela, örnekte yer alan üç karakterlik [7,4,C(a)] kodu, 7 karakter geri git, 4 karakter kopyala, sonuna *a* karakterini ekle şeklinde açılır.

Uygulanması kolay olan bu algoritmanın en büyük dezavantajı, eğer arama tamponunda aranan tekrarlar bulunamazsa kullanılan üçlü sistemin 1 byte'lık veriyi [0,0,C(a)] şeklinde temsil etmesi nedeniyle sıkıştırma yerine genişletme yapmasıdır. LZ77 algoritması temel alınarak geliştirilen, LZSS olarak adlandırılan bir yaklaşım, bu israfı tek bitlik bir bayrak kullanarak ortadan kaldırmıştır. Bu bayrak kendisinden sonra gelen verinin tek bir karakter mi yoksa bir karakter katarını ifade eden veri mi olduğunu belirler. Bu bayrak sayesinde üçüncü elemana da gerek kalmamıştır.

Eğer arama tamponu boyutu yeteri kadar büyük değilse tekrarlar bulunamaz. Örnek te arama tamponunun boyutunu 6 olarak kabul edilseydi sıkıştırılacak benzerlik

bulunamazdı. Öte yandan arama tamponu çok büyükse, arama zamanı ve dolayısıyla da sıkıştırma zamanı artar. Her ne kadar geliştirilen efektif arama yaklaşımları ile bu zaman bir ölçüde kısaltılabilmişse de, arama tamponu yine de çok büyük seçilmemelidir.

LZ77 yaklaşımının tıkandığı bir nokta vardır. Eğer periyodik bir dizimiz varsa ve periyodu arama tamponundan büyükse, hiç benzeşme bulunamaz ve her karakter için fazladan gönderdiğimiz veriler nedeniyle sıkıştırma yerine genişletme yapmış oluruz. Periyodik olarak tekrarlanan 8 karakter uzunluğundaki "abcdefgh" karakter grubu için benzeşme bulunamayacağı Şekil 4.5' de görülmektedir.



Şekil 4.5. LZ77'in tıkandığı periyodik tekrarlamaya örnek

4.5.3.2. LZ78 sıkıştırma algoritması

Lempel ve Ziv, LZ77'de yer alan tampon büyüklüğünün sınırlı olmasının yarattığı sıkıntıları ortadan kaldırmak için, tampon kullanmayan çok farklı bir yöntem geliştirerek 1978'in Eylül ayında yayınlanan "Compression of Individual Sequences via Variable-Rate Coding" isimli makalelerinde bu algoritmalarına yer vermiştir [25]. LZ77'den çok farklı bir yapıda olması nedeniyle, bu algoritma ve geliştirilmiş biçimleri, LZ78 (veya LZ2) ailesi olarak adlandırılmıştır. LZ77'den farklı olarak, sadece metin tabanlı sıkıştırmada değil, bitmap gibi farklı sayısal veriler üzerinde de başarıyla uygulanabilmiştir [5].

LZ78 sıkıştırma algoritması, hem kodlayıcı (encoder) hem de çözücü (decoder) tarafından aynı sözlüğün oluşturulması prensibine dayanır. Kaynaktan okunan sembol grupları sözlükte bulunan en uzun ön eklerinin indeksi ile birleştirilerek

ikililer (pairs - [*i*,*c*]) ile kodlanır. Bu ikilideki *i*, yeni girişin sözlükte bulunan en uzun ön ekinin indeksi, *c* ise, bu ön eki takip eden karakterdir.

LZ78 açma algoritması, sıkıştırma algoritmasında oluşturulan sözlüğe ilk etapta sahip değildir. Bir yandan açma işlemi yapılırken diğer yandan sözlük oluşturulur. Okunan ikili kodun ilk karakteri 0 ise, bu ikilinin aslında tek karakteri temsil ettiği anlaşılır ve ikinci karakter okunarak kodlanır ve sözlüğe eklenir. İlk karakter sıfırdan farklı ise, bu kod iki ya da daha fazla karakterin birleşmesi ile oluşan bir grubu temsil ediyordur. Bu durumda, sözlüğe eklenmiş olan elemanlar içinde bu kodun iki elemanının da karşılıkları bulunur ve kodlanır.

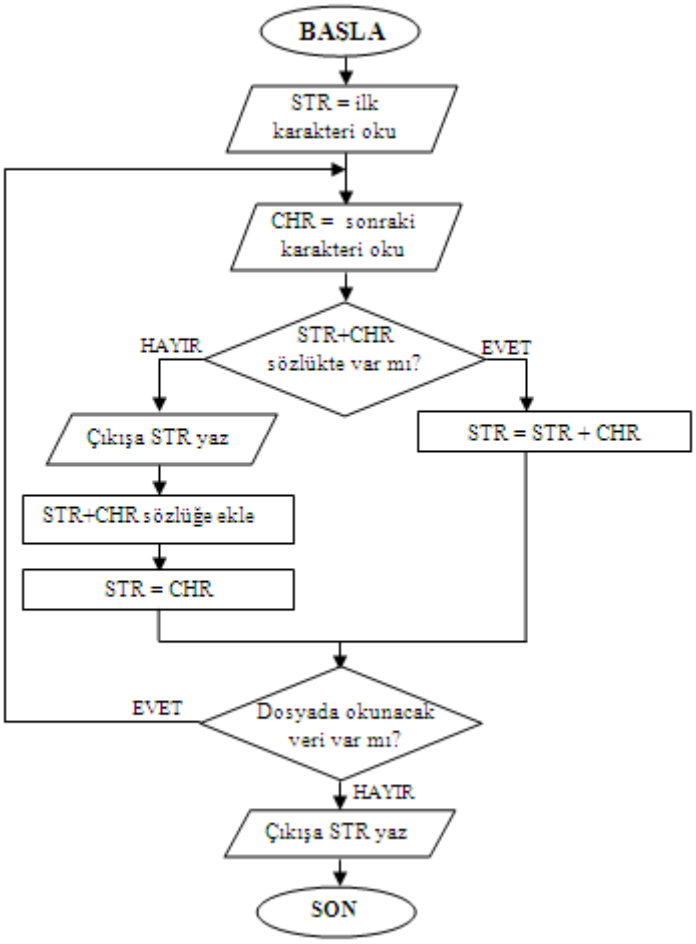
LZ78 algoritması kalıpları bulma ve ayrı ayrı saklama becerisine sahip olsa da, en önemli dezavantajı, sözlüğün sınırsız bir şekilde büyümesidir. Pratikte, sözlüğün büyümesi belirli bir noktada durdurulmalı, ya gereksiz girdiler elenmeli, ya da kodlama sabit sözlük şemasına zorlanmalıdır.

4.5.3.3. LZW sıkıştırma algoritması

Terry Welch 1984'te Unisys için çalışırken, LZ78 yaklaşımını yüksek performanslı disk ünitelerine uyarlamış ve ortaya çıkan yeni algoritma LZW olarak kabul görmüştür. LZW hem sıkıştırma hem de açma performansı açısından LZ78 ailesinin en iyisi olmayı başarmıştır [26]. Her tip veri üzerinde iyi sonuçlar veren bir algoritma olduğu için, kendisinden sonra gelen birçok algoritma LZW'yi temel almıştır. 1985 yılından beri Unisys LZW'nin patentini elinde bulundurmaktadır. UNIX işletim sistemindeki "compress" emri de bu yönteme göre çalışmaktadır.

LZW sıkıştırma algoritmasında, LZ78'de kullanılan ikili yapısındaki ikinci elemanın gerekliliği ortadan kalkmıştır. Kodlayıcı, önce kaynaktaki tüm karakterlerden bir sözlük oluşturarak gönderir. Bu karakterler bir ön geçişle bulunabilir. Eğer ASCII tablosundaki tüm karakterler kullanılacaksa ön geçiş yapılmasına ve sözlüğün gönderilmesine gerek yoktur. Kodlama aşamasında okunan her karakter sözlükte

aranır. Bulunursa bir sonraki karakter de okunur ikisi birleştirilerek aranır. Sözlükte karşılığı bulunamayana kadar bu şekilde devam eder. Sözlükte karşılığı bulunmayan bir girdiye ulaşıldığında ise son karakteri hariç önceki karakterlerinin sözlükteki karşılığı kodlanır. Bu kod ile son karakterin sözlükteki kodu birleştirilerek sözlükte yeni bir girdi oluşturulur. Son karakter, sonraki adımda ilk karakter yapılarak kodlamaya devam edilir.



Şekil 4.6. Lzw Sıkıştırma Algoritması

Şekil 4.6'da verilen sıkıştırma algoritmasında da görüldüğü üzere, veriler giriş dosyasından byte byte okunup bir stringe eklenir ve bu stringin tabloda mevcut olup olmadığına bakılır. Eğer mevcut ise, dosyadan bir byte daha okunarak işlem tekrarlanır. Algoritma, sıkıştırılacak olan metin üzerinde, sözlükte olan bir kelimeyle uyuşan harfler bulduğu sürece ilerler. Farklı bir harfe rastlandığı zaman, o ana kadar

uyumlu bulduğu harflerden oluşan kelimenin kodunu sonuca yazar ve yeni harfi içeren kelimeyi sözlüğe ekler.

LZW yöntemi ile sıkıştırma yapılırken, başlangıçta ilk 256 elemanı (0-255) ASCII tablodaki karakterler için ayrılmış olan "dictionary" adı verebileceğimiz bir tablo tutulur. Sıkıştırma sonrası çıkışa yazılacak veriler bu sözlükten alınacak indis numaraları olduğu için, çıkış dosyasına yazılacak her bir verinin boyu en az 9 bit (8 bitten fazla) olmak zorundadır. 12 bit için çıkış dosyasına yazılabilecek kodlar 256-4095 arası değişir. 4095'ten sonra gelen veriler kodlanmaksızın çıkış dosyasına yazılır. Bu sıkıştırma oranını bir miktar düşürür [27].

Örnek: "erken_ erken _ erken _ erken _ erken _ erken _ erken" karakter serisini LZW ile kodlayalım.

Başlangıçta sözlükte tüm ASCII karakterleri bulunacaktır. Tablo 4.1' de sadeleştirme yapılarak sadece seride kullanılan karakterlere yer verilmiştir. Çizelgenin ilk 5 elemanı bu karakterlerden oluşmaktadır. 256 ve daha sonraki karakterler, LZW tarafından kodlama yapılırken sözlüğe eklenen karakterlerdir.

Tablo 4.1. LZW ile oluşturulan sözlük yapısı

Sözlük Sırası	Temsil Ettiği	Sözlük Sırası	Temsil Ettiği	Sözlük Sırası	Temsil Ettiği	Sözlük Sırası	Temsil Ettiği
95	_	258	ke	265	erke	272	rken
101	e	259	en	266	en_	273	n_erk
114	r	260	n_	267	_er	274	ken_
107	k	261	_e	268	rke	275	_erk
110	n	262	erk	269	en_e		
256	er	263	ken	270	erken		
257	rk	264	n_e	271	n_er		

Kodlama aşaması şu şekilde gerçekleşir: Önce ilk iki karakter *e* ve *r* karakterleri okunur. İki karakterin birleşimi olan *er* henüz sözlükte olmadığı için, [101,114] kodu ile sözlüğe 6. eleman olarak eklenir. Tablo 4.1' de sözlükteki gerçek kod değerlerine değil, sadece temsil ettikleri karakter öbeklerine yer verilmiştir. İlk karakter sözlükteki karşılığı olan 101 ile kodlanarak ikinci karakter ilk karakter yapılır ve

kodlamaya devam edilir. Sözlüğün 261'inci karakterine kadar kodlama benzer şekilde olacaktır. Serideki ikinci *erken* kodlanırken ilk *er* daha önce sözlüğe eklenmiş olduğu için bu defa *e* karakterinin kodu olan 101 değil, *er* ikilisinin kodu olan 256 kodlanır. Sözlüğe eklenirken ise bir sonraki karakter olan *k* ile birleştirilerek *erk* karakter öbeği oluşturulur ve 262'inci sözlük girdisi [256,107] kodu ile eklenir. Kaynak bitene kadar bu şekilde kodlanacak ve sözlük büyümeye devam edecektir. Kaynak verinin LZW ile kodlanmış hali aşağıda verilmiştir:

101,114,107,101,110,95,256,258,260,262,259,261,257,266,265,264,268,271,263,267,263

Serinin tamamı kodlandığında Tablo 4.1'de verilen 275 elemanlı sözlük oluşacaktır. Bu sözlük sıkıştırma ve açma aşamalarında bellekte oluşturulur. ISSDC'de olduğu gibi asıl veriden önce gönderilen ve dosya içinde saklanan bir sözlük değildir. Bu nedenle LZW'de aslında sözlük masrafı yoktur.

LZ78'de olduğu gibi LZW'de de açma algoritması, sıkıştırma algoritmasında oluşturulan sözlüğe ilk etapta sahip değildir. Sözlük sıkıştırma algoritmasında yaratıldığı gibi açma algoritmasında da yaratılır. Örnekte üretilen çıktıyı düşünelim. İlk altı karakter (101,114,107,101,110,95) *erken_* karakter öbeğinin ASCII kodlarıdır. İlk adımda 101 ve 114 bir araya getirilip sözlüğün 256'ıncı elemanı, yani *er* oluşturulur. Birer karakter kaydırma ile sırasıyla [114,107] ile 257, [107,101] ile 258, [101,110] ile 259, [110,95] ile 260 ve [95,101] ile 261'inci sözlük girdileri oluşturulur. Kod çözücü kodlanan verinin yedinci karakteri olan 256'ya ulaştığında, artık bu kodun karşılığını sözlükten bulabilir. Bir yandan açma işlemi sürerken, diğer yandan sözlük 275'inci girdiye kadar benzer şekilde oluşturulur.

LZ ailesinin en çok kullanılan üyesi LZW'dir. Hızı ve sıkıştırma performansı yüksektir. Fakat, LZ78'de olduğu gibi, LZW algoritmasında da sözlük büyüklüğünün sürekli artması sorun yaratmaktadır. Çözüm olarak birçok farklı yöntem geliştirilmiştir. LZW tabanlı çalışan bir kayıpsız görüntü sıkıştırma algoritması olan Gif ve Unix'teki "compress", sözlük büyüklüğü 2^b değerine eriştiğinde (b değeri

önceden belirlenir), sözlüğü iki katına (2^{b+1}) çıkartır. Sözlük daha da büyüdüğünde, statik sözlük yaklaşımına döner. Eğer sıkıştırma oranı belli bir seviyenin altında kaldıysa sözlüğü boşaltarak, sözlük oluşturma işlemini yeniden başlatır [28].

Tablo 4.2. LZW algoritması ile verinin sıkıştırılma aşamaları

	Sıkıştırılmış Çıktı	Sözlük	Tampon	Sıkıştırılmamış Veri
a)			0	100110101
b)	0	2(0,1)	0	100110101
c)	01	3(1,0)	1	00110101
d)	010	4(0,0)	0	0110101
e)	010		0	110101
f)	0102	5(0,1,1)	2	10101
g)	0102		1	0101
h)	01023	6(1,0,1)	3	101
i)	01023		1	01
j)	01023		3	1
k)	010236		6	

Tablo 4.3. LZW algoritması ile verinin sıkıştırma açma aşamaları

	Sıkıştırılmış Çıktı	Sözlük	Tampon	Sıkıştırılmamış Veri
a)	0			010236
b)	01	2(0,1)	0	10236
c)	010	3(1,0)	1	0236
d)	01001	4(0,0)	0	236
e)	0100110	5(0,1,1)	2	36
f)	0100110101	6(1,0,1)	3	6

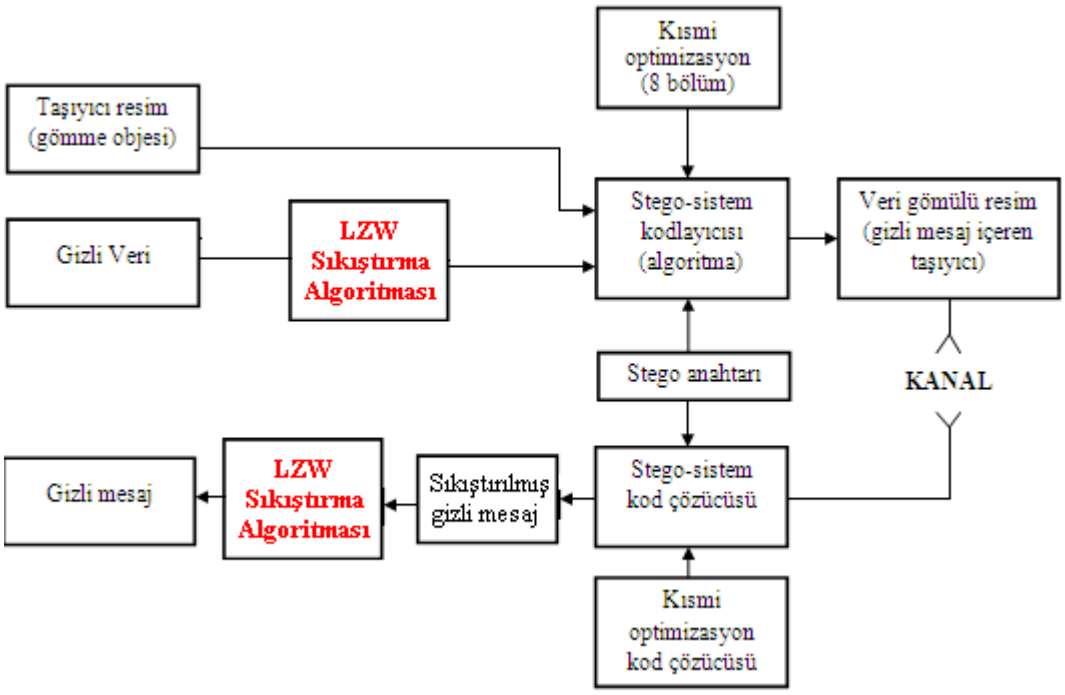
Tablo 4.2 ve Tablo 4.3'te bir verinin sıkıştırma ve sıkıştırma açma işlemi süreci tablosal olarak gösterilmiştir. Tablo 4.2 incelendiğinde 10 bitlik bir verinin lzw algoritmasıyla 6 bitlik bir veri haline dönüştüğü ve bu süreç içerisinde sözlükte ve cıkışta alınan değerler görülebilir. Bu işlemle %40 lık bir sıkıştırma sağlanmıştır.

Tablo 4.3’de ise sıkıştırılmış 6 bitlik verinin sıkıştırma açma işlemine tabi tutulması sonucu, orjinal verinin elde edildiği görülmektedir.

BÖLÜM 5. LZW SIKIŞTIRMA ALGORİTMASI TABANLI VERİ GİZLEME UYGULAMASI

Veri gizleme uygulamalarında, resimde yer alan orijinal piksellerin korunması büyük önem arz etmektedir. Çünkü gizleme amaçlı yapılan bu uygulamanın dışarıdan algılanmaması uygulamanın temel amacını oluşturmaktadır. Bu nedenle kısmi optimizasyon tekniğinin sıkıştırma algoritmasıyla desteklenmesinin, veri gizleme işleminde daha efektif sonuçlar elde edilmesini sağlayacağı öngörülmüştür. Bunun için optimizasyon ile gizlenecek olan verinin, kayıpsız ve dinamik sözlük tabanlı bir sıkıştırma tekniği olan LZW sıkıştırma algoritması ile sıkıştırılarak, boyutunun minimize edilmesi temel alınmıştır. Bu sayede optimizasyon ile veri gizleme işlemi sonucu örtü verisinde meydana gelen bozulmanın en aza indirgenmesi ve veri gizleme işleminin daha kısa süre içerisinde gerçekleştirilmesi sağlanmıştır.

Bu sistemde, gömme objesi olarak .bmp uzantılı görüntü verisi kullanılmaktadır. Bu görüntü içerisine gizlenmek istenen mesajın boyutu, Şekil 5.1'de de görüleceği üzere lzw sıkıştırma algoritmasına tabi tutularak minimize edilir ve elde edilen yeni veri .lzw uzantısıyla oluşturulur. Kısmi optimizasyonun temel kodlama bölümü olan stego sistem kodlayıcısına gömme objesi ve sıkıştırmayla elde edilen .lzw uzantılı veri gönderilir. Böylece kodlama sonucunda gömme objesi üzerinde oluşacak hata bitlerinin sayısı sıkıştırma oranında azalır. Kodlama işlemi sonucunda stego anahtarı kullanıcı tarafından girilir ve kodlama tamamlanır. Bu işlem sonucunda oluşan, veri gömülü resim içerisinden gizlenmiş verinin çıkartılması işleminde ise kodlamada kullanılan aynı anahtar kullanılmaktadır. Stego-sistem kod çözücüsü ile gömme objesi içerisinden gizlenmiş veri elde edilir. Bu veri .lzw uzantılı sıkıştırılmış dosyadır. Bu dosya sıkıştırma açma algoritmasına tabi tutularak orijinal gizlenmiş veri elde edilir.

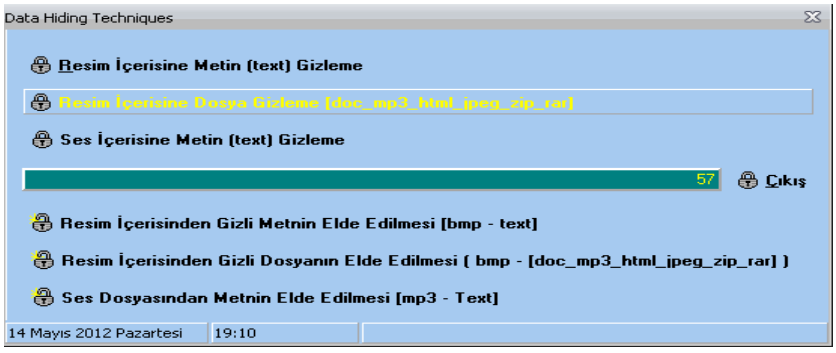


Şekil 5.1. Lzw tabanlı kısmi optimizasyon stenografik sisteme ait kodlayıcı ve kod çözücü blok diyagramı

5.1. Uygulama Yazılımının Tanıtılması

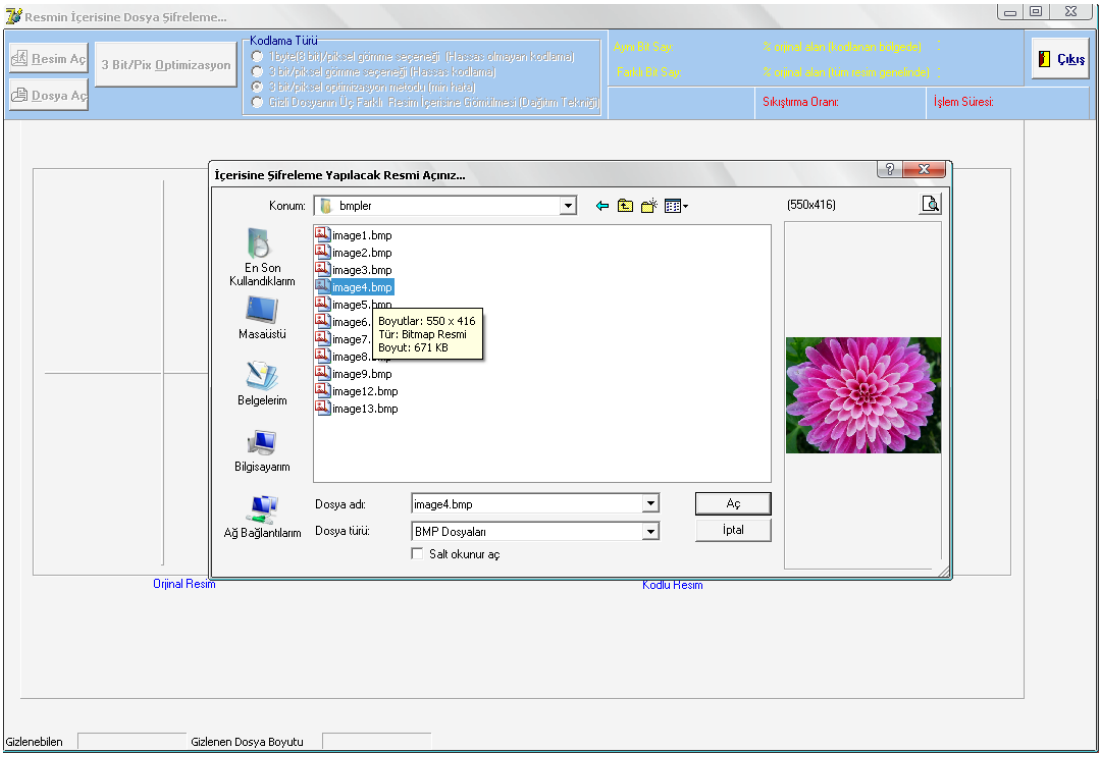
Kısmi optimizasyon tekniği ile veri gizleme/çıkarma uygulamasını içeren uygulama yazılımının çalıştırılabilir program dosyası Delphi 7.0 program sürümü ile tasarlanmıştır. Yaklaşık olarak 10000 satırdan oluşmakta olup 4,5 MByte büyüklüğündedir. Bu tez çalışmasında, veri gizleme uygulamasına, yine Delphi platformunda tasarlanmış, yaklaşık 1000 satırdan oluşan lzw sıkıştırma algoritması eklenmiştir.

5.1.1. Resim içerisine dosya gizleme uygulaması



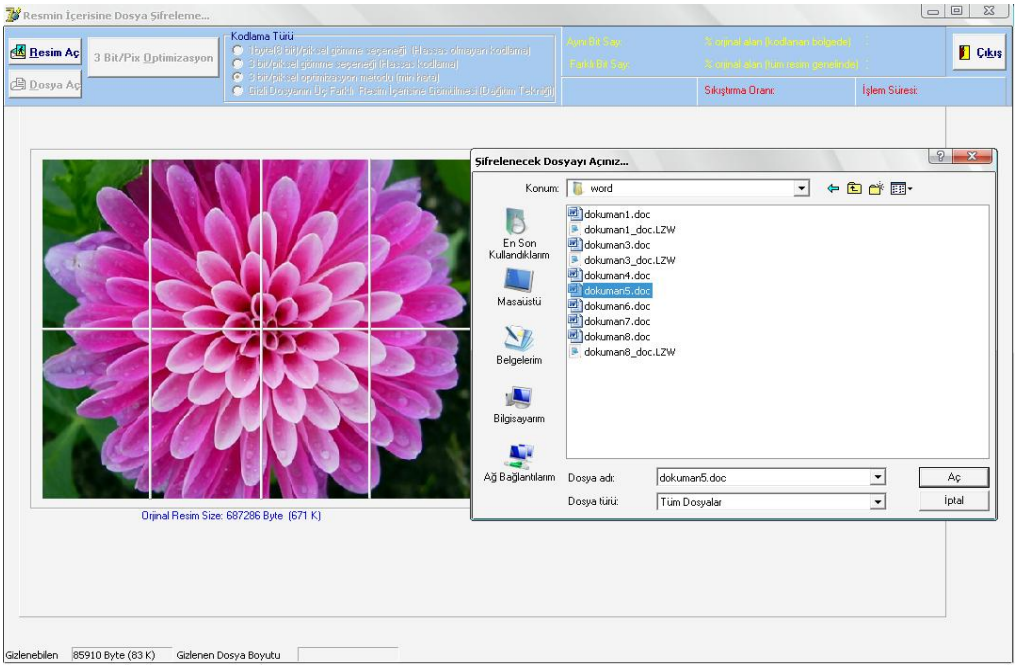
Şekil 5.2. İki ana bölüm ve altı alt daldan oluşan şifreleme / şifre çözme programı arabirimi

Şekil 5.2'de görülen uygulamaya giriş ekranında, üstteki bölüm veri gizleme, alttaki bölüm ise gizlenen veriyi elde etme uygulamalarına giriş bağlantılarını içermektedir. Kısmi optimizasyon ile veri gizlenmek istendiğinde, bu uygulamayı içeren, Şekil 5.2'de de belirtilen 'Resim İçerisine Dosya Gizleme' butonu tıklanarak uygulamaya geçilebilir.



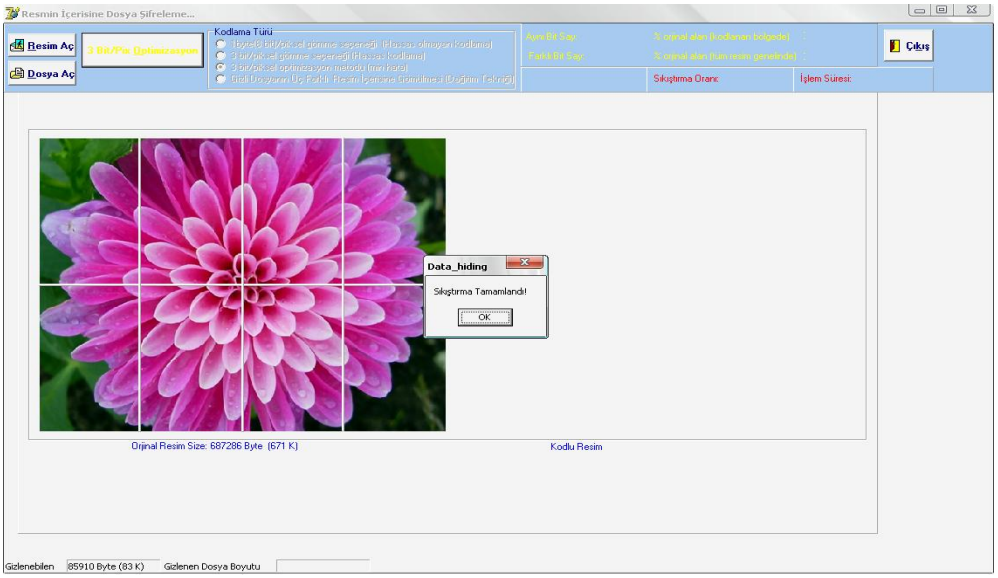
Şekil 5.3. Optimizasyon tekniği ile gizli veri gömme uygulamasında gömme objesi seçimi

Şekil 5.3'te optimizasyon tekniğini kullanarak veri gizleme yapılacak örtü verisinin seçim aşaması gösterilmektedir. Kısmi optimizasyon yöntemi kullanılmak isteniyorsa, kodlama türü bölümünde "3 bit/piksel optimizasyon metodu" seçeneği işaretlenir ve "Resim Aç" butonu ile istenilen .bmp uzantılı gömme objesi(örtü verisi) belirlenir.



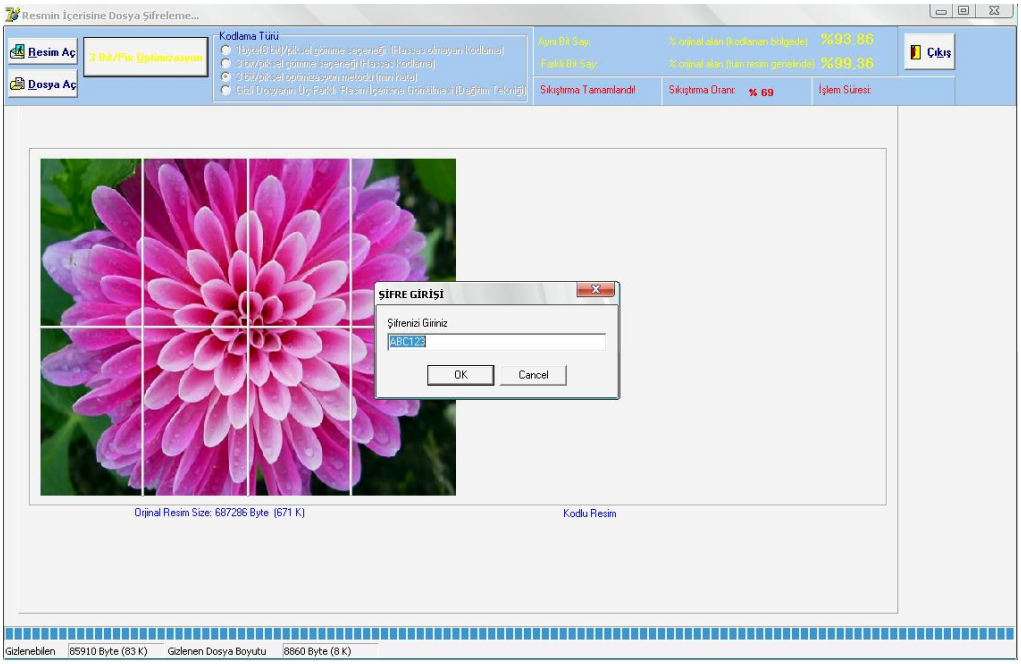
Şekil 5.4. Optimizasyon tekniği ile gizli veri gömme uygulamasında gizlenecek verinin seçimi

Şekil 5.4'te ise seçilen gömme objesi içerisine gizlenmek istenen veri seçimi yapılmaktadır. "Dosya Aç" butonuna tıklayarak açılan pencerede dosya seçimi yapılır. "3 Bit/Pix Optimizasyon" butonuna tıklanmasıyla öncelikle gizlenecek olan veri sıkıştırma işlemine tabi tutulur. Verinin boyutu minimize edilir ve ekrana Şekil 5.5'de görülen "Sıkıştırma Tamamlandı!" uyarı mesajı gelir. Devamında optimizasyon ile veri gömme işlemi başlatılır.



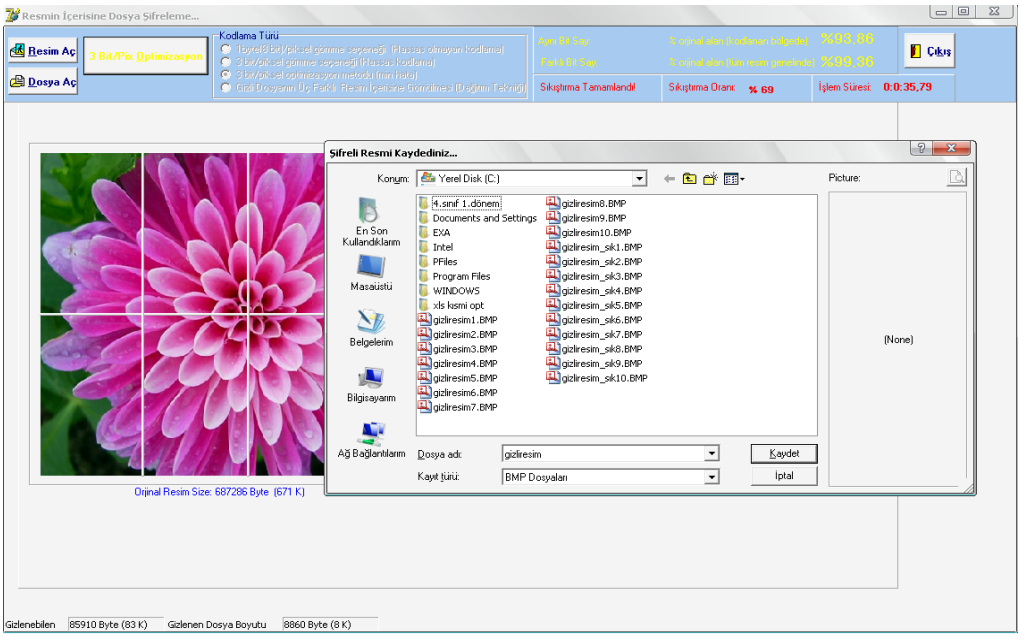
Şekil 5.5. Optimizasyon tekniği ile gizlenecek verinin sıkıştırılması

Veri gizleme algoritmasının tamamlanması sonucu stego anahtar olarak adlandırdığımız şifre kullanıcı tarafından girilir (Şekil 5.6).

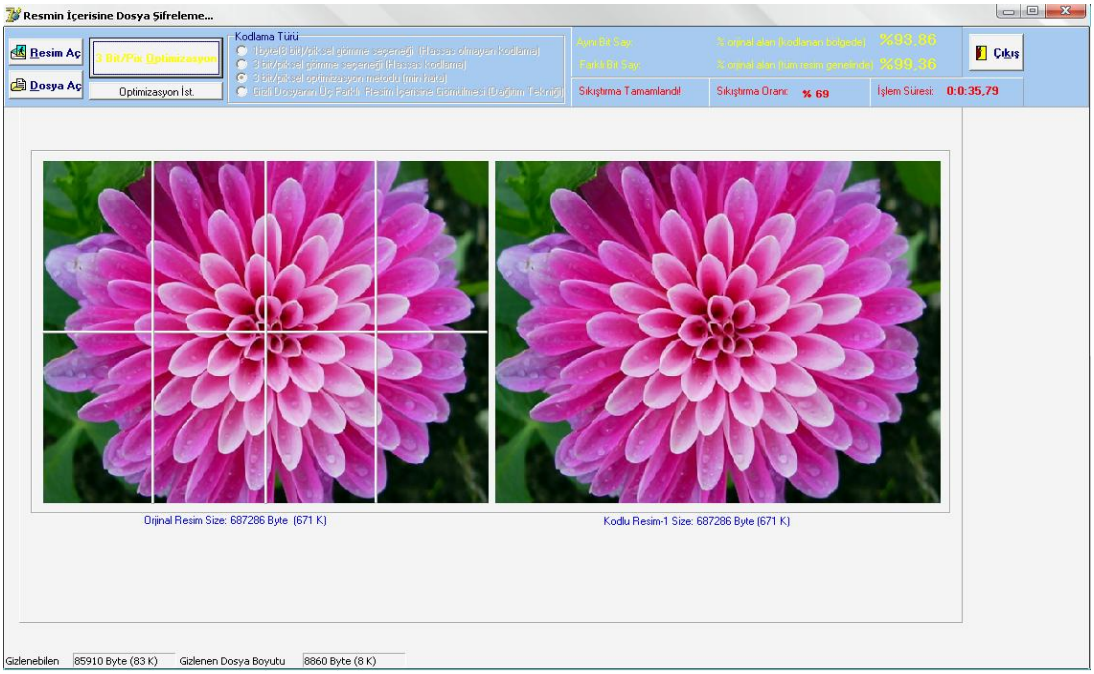


Şekil 5.6. Optimizasyon tekniği ile veri gizlemede şifre girişi

Şifre girişi tamamlandığında, Şekil 5.7'de görüldüğü gibi içerisine veri gizlenmiş olan resim istenilen konuma, istenilen isim ile kaydedilir.

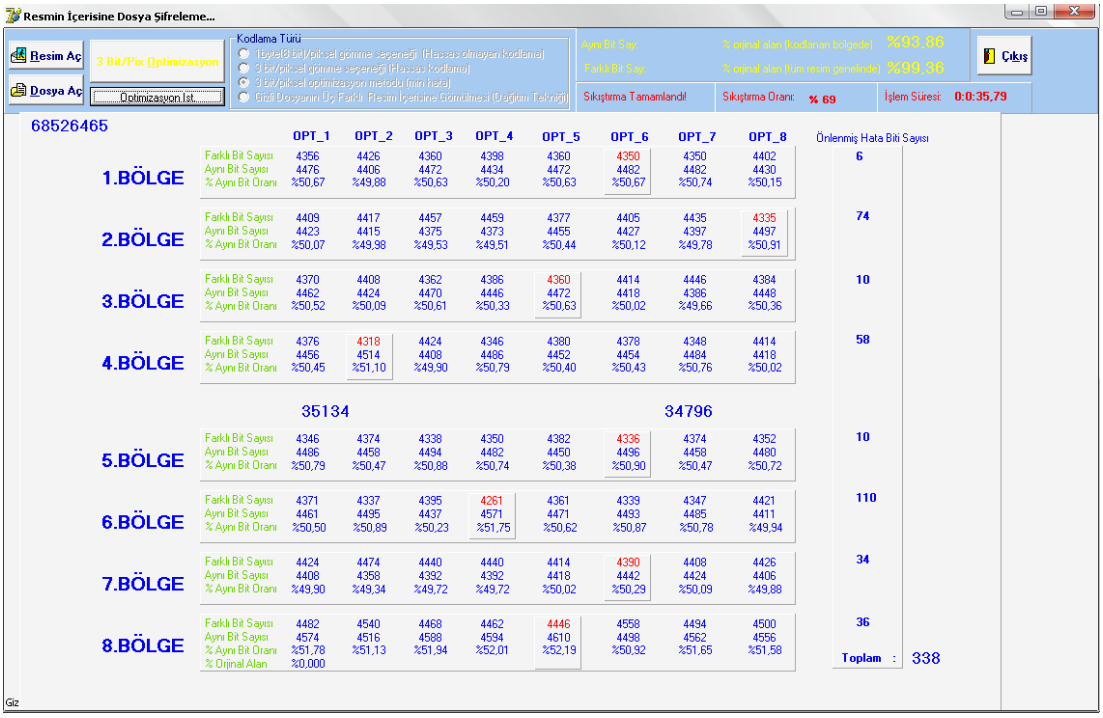


Şekil 5.7. Optimizasyon tekniği ile gizlenmiş verinin kaydedilmesi



Şekil 5.8. Optimizasyon tekniği ile veri gizleme sürecinin tamamlanması

Şekil 5.8 incelendiğinde, sol tarafta orijinal gömme objesi, sağ tarafta ise içerisine veri gizlenmiş olan görüntü görülmektedir. İki resim arasındaki fark insan gözü tarafından algılanamamaktadır. Burada yer alan veriler incelenecek olursa; gizlenecek olan verinin %69 oranında sıkıştırılarak boyutunun 8860 bayta indirgendiği görülmektedir. Ayrıca işlem süresi 35,79 sn'dir. Yapısal olarak resmin 8 farklı bölgeye ayrılarak her bir bölgeye 8 farklı yöntem denenerek, bölgesel bazda en optimum sonucun elde edilmesi için gerekli olan işlem süresi diğer tekniklerle karşılaştırıldığında doğal olarak 8 kat daha uzun sürecektir. Fakat bu sorun sıkıştırma algoritması desteğiyle en aza indirgenmiştir.



Şekil 5.9. Optimizasyon tekniği ile veri gömme sonucu istatistikler

Şekil 5.9'da, "Optimizasyon İst." butonuna tıklanarak, gömme sonucunda elde edilen istatistiksel verilerin sergilendiği pencere görülmektedir. Burada standart metot (opt_1) ile optimizasyon tekniği arasında sayısal bir değerlendirme yapılmaktadır. Buradan anlaşılmaktadır ki her bölgede elde edilen bozulmamış (orijinalliği korunan) bit sayısı bakımından diğer metoda göre toplamda orijinal piksellerin 338 bit daha fazla olduğu ispatlanmıştır.

Klasik LSB veri gömme tekniği bölgesel veri gömme tekniğinin ilk sırasını kullanmakta olup bu tekniğin tüm resme uygulanması durumunda bu örnek uygulama için 35134 hatalı bit oluşacaktır. Ancak kısmi optimizasyon metodu ile sekiz bölgeye sekiz farklı yaklaşım uygulandığından her bölgede en az hata/bit oranını veren veri gömme tekniği belirlenerek orijinal bit oranı maksimize edilmektedir. Bu örnekte kazanılan toplam bit sayısı 338 olmaktadır. Çünkü hata bitlerinin sayısı 34796 olarak oluşmuştur. Böylelikle sıkıştırma tabanlı kısmi optimizasyon metodu ile en az hata/bit oranına sahip en uygun alternatif çözüm elde edilmiştir. Bu kural, resmi oluşturan tüm bölgelere uygulanmak suretiyle her bir bölgeden elde edilen kazanımlar (orijinalliği korunan pikseller) ile toplamda da en az

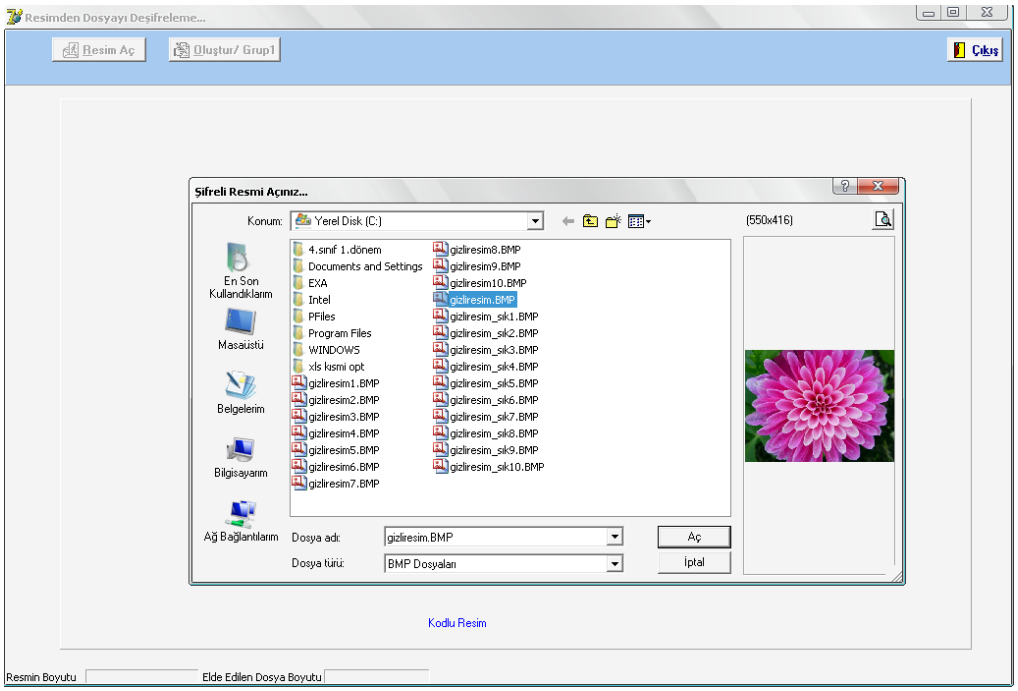
hata bitinin oluşmasını sağlamaktadır. Bu örnekte kısmi tarama neticesinde seçilen optimizasyon sonuçları aşağıda verilmiştir:

Birinci bölge	→	Opt_6
İkinci bölge	→	Opt_8
Üçüncü bölge	→	Opt_5
Dördüncü bölge	→	Opt_2
Beşinci bölge	→	Opt_6
Altıncı bölge	→	Opt_4
Yedinci bölge	→	Opt_6
Sekizinci bölge	→	Opt_5

5.1.2. Resim içerisinden gizli dosyanın elde edilme uygulaması

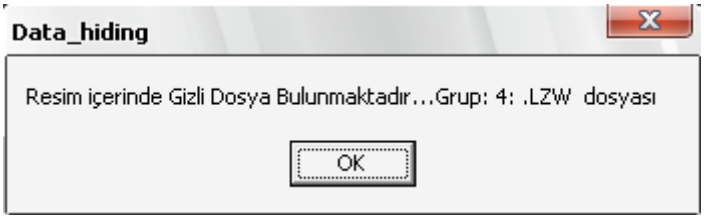
Sıkıştırma tabanlı kısmi optimizasyon tekniği ile gömme işlemi tamamlanan resim içerisinden gizli dosyanın yeniden elde edilmesi için Şekil 5.10'da görülen veri gizleme yazılımı arabiriminde ikinci bölüm, ikinci buton tıklanır.

Şekil 5.10. İki ana bölüm ve altı alt daldan oluşan şifreleme / şifre çözme programı arabirimi



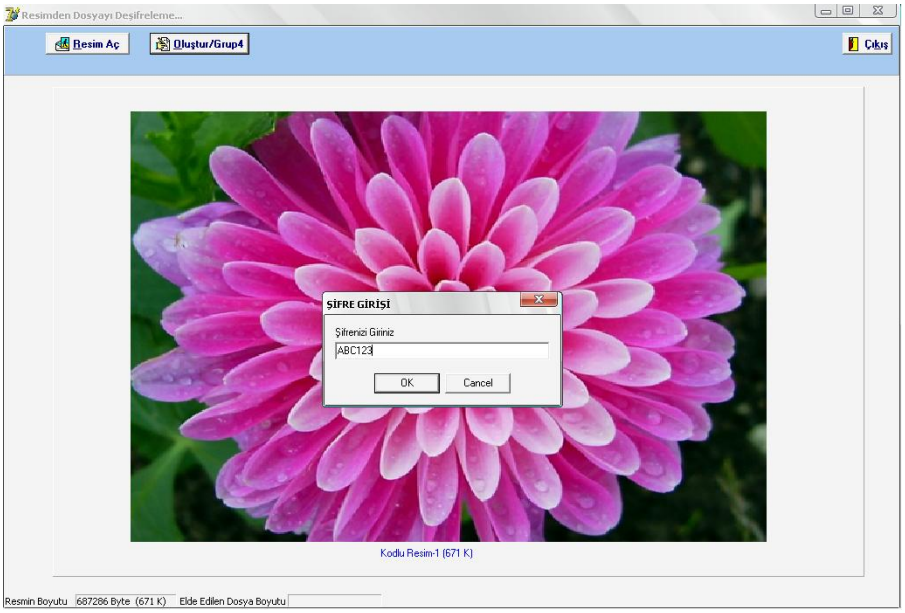
Şekil 5.11. İçerisinde gizli veri olan görüntü dosyasının seçimi

Klasik olarak “Resim Aç” butonu ile gizli dosya gömülü resim seçilerek aç butonu ile resim dosyası taranarak içerisinde gizli veri olup olmadığı araştırılır(Şekil 5.11).



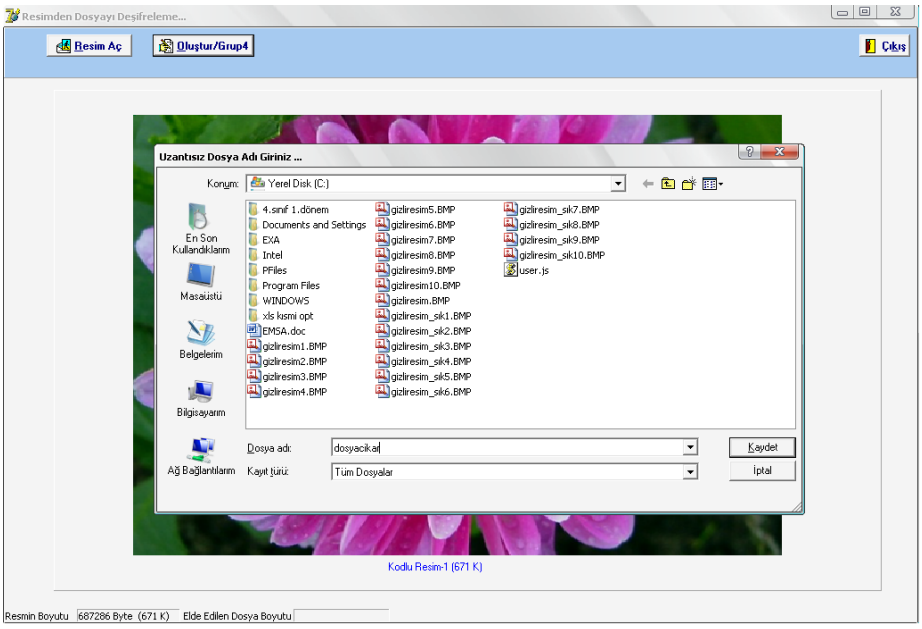
Şekil 5.12. İçerisine veri gömülü olan görüntünün tarama sonucu

Şekil 5.12’de görüldüğü gibi “Grup 4” olarak adlandırılan veri gömme tekniği - ki bu teknik optimizasyon yöntemini içermektedir - tespit edilerek veri türü de sıkıştırılmış dosya biçimi olan “lzw” olarak bildirilmektedir.



Şekil 5.13. Çözme işlemi öncesi şifre sorgulaması

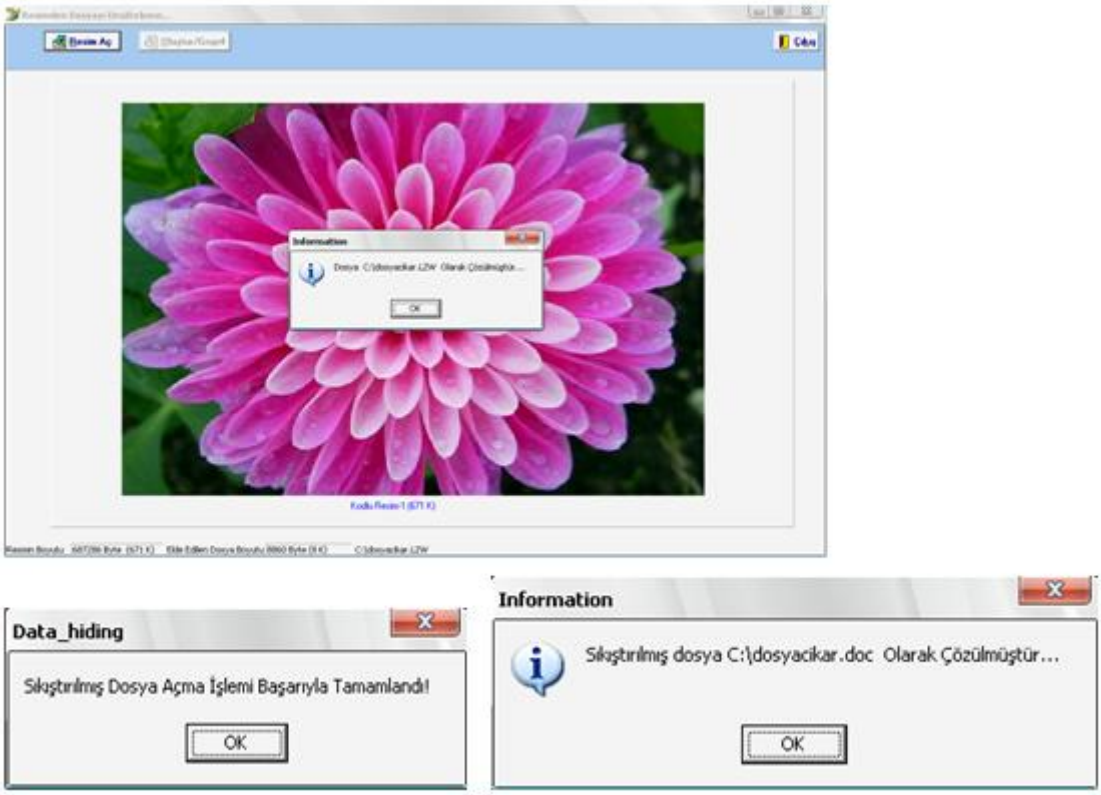
Resim açıldıktan sonra “Oluştur/Grup 4” butonu tıklanarak resim içerisindeki verin elde edilmesi için ekrana gelen şifre sorgu penceresinde, verinin gizlenmesinde kullanılan şifre doğru biçimde girilir ve çözme işlemi başlatılır (Şekil 5.13).



Şekil 5.14. Çözme işlemi sonucu oluşturulacak dosyanın seçilmesi

Şifre giriş işleminden sonra ekrana, elde edilen gizli verinin bilgisayarda nereye kaydedileceğini soran bir iletişim penceresi gelir (Şekil 5.14). Bu pencerede elde

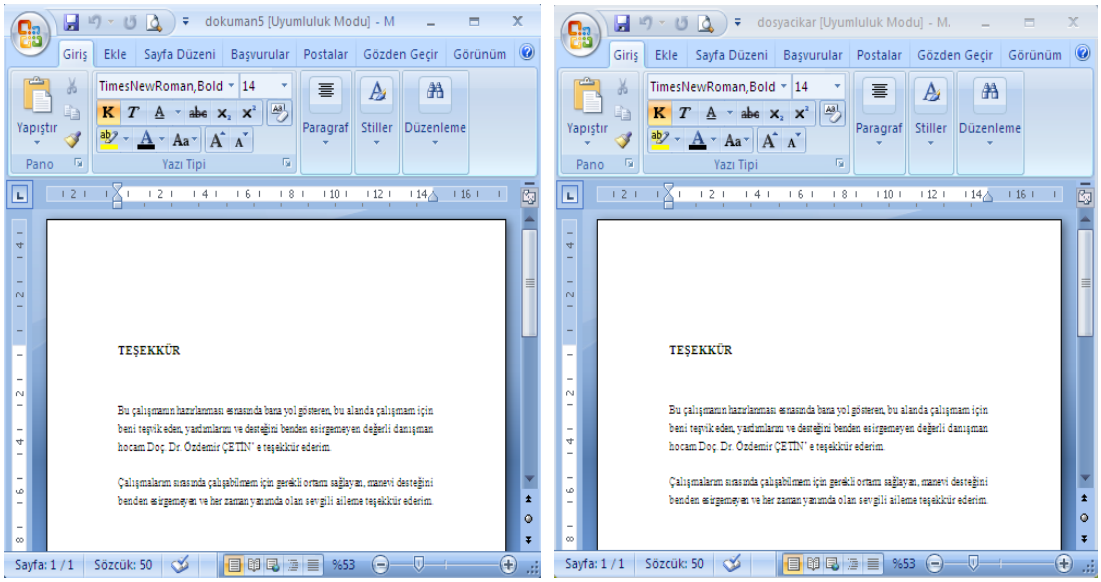
edilen gizli veriye istenilen bir isim verilerek bilgisayarda istenilen bir yere kaydedilir.



Şekil 5.15. Çözme işlemi ile elde edilen sıkıştırılmış veriden orijinal verinin elde edilmesi

Çözme işlemi tamamlandığında belirtilen konum ve isimde lzw dosyası oluşmuş olur. Ancak lzw dosyası asıl ulaşılmak istenen veri değildir. Sıkıştırılmış veri olan lzw dosyası, otomatik olarak sıkıştırma açma işlemine tabi tutulur ve aynı konumda orijinal uzantısıyla birlikte oluşturulur (Şekil 5.15).

Sekil 5.16'da orijinal gizli veri ile elde edilen gizli veri gösterilmektedir. Şekilde sağ tarafta görülen geri elde edilen bilgi, sol tarafta görülen ise saklanan bilgidir.

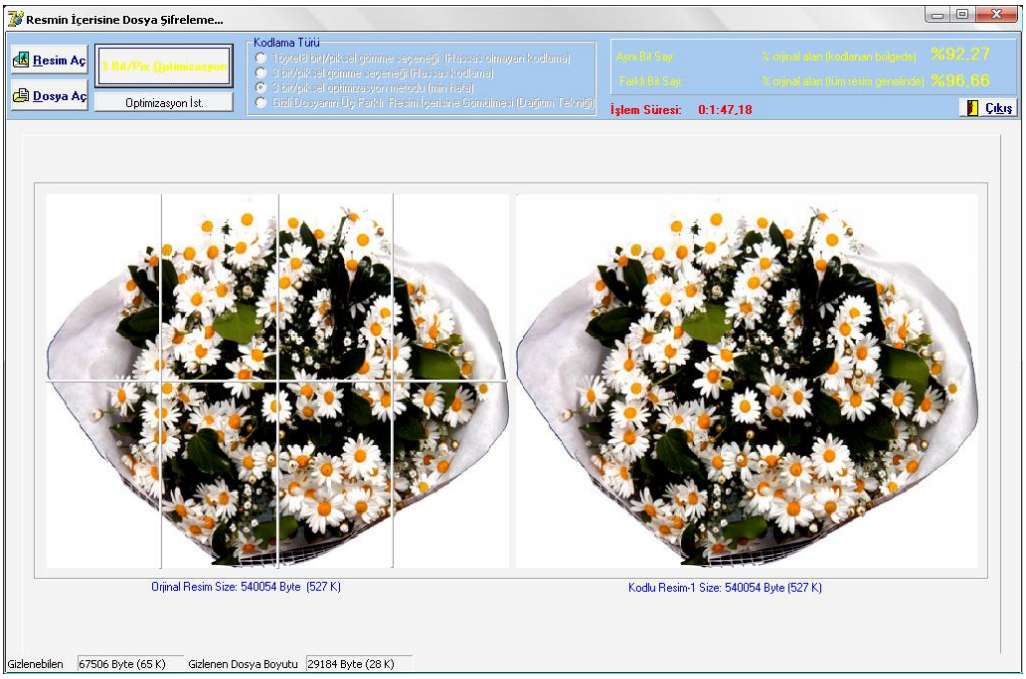


Şekil 5.16. Orijinal gizli veri ve elde edilen gizli veri

5.2. Uygulamaya Ait Deneysel Sonuçlar

Kısmi optimizasyon ile veri gizleme tekniğinin, lzw sıkıştırma algoritmasıyla birlikte geliştirilmesiyle elde edilen sistemin getirdiği katkının ölçülmesi için farklı veriler üzerinde deneysel bir çalışma gerçekleştirilmiştir. Bu çalışmada, önerilen sıkıştırma tabanlı optimizasyon tekniğinin algılanabilirlik ve gizli veri gömme süreleri gibi kriterlere bağlı başarımları değerlendirilmektedir.

Birbirinden farklı boyutlarda, bmp uzantılı görüntü verileri ile yine birbirinden farklı tür(doc, xls, tif, bmp, htm, ppt, txt) ve boyutlarda veriler, kısmi optimizasyon yöntemi ile gizleme işlemine tabi tutularak elde edilen sonuçlar ile aynı verilerin lzw tabanlı uygulama ile gizleme işlemine tabi tutulmasıyla elde edilen sonuçların karşılaştırılması yapılmıştır. Bu karşılaştırmada, gizleme işlemi sonucu görüntü verisi üzerinde oluşan hatalı bit sayısı, işlemin tamamlanması için geçen süre ve sistemin LSB yöntemine göre koruduğu orijinal bit sayısı esas alınmıştır.



Şekil 5.17. Kısmi optimizasyon yöntemi ile veri gömme uygulaması

Şekil 5.17'de görülen veri gizleme uygulamasında, 500x360 boyutunda 540054 bayt(527 K) büyüklüğünde bir örtü verisi içerisine 29184 bayt(28 K) büyüklüğünde .doc uzantılı bir metinsel veri kısmi optimizasyon tekniği ile gizlenmiştir. Gizleme işlemine ait istatistiksel veriler Tablo 5.1'de görülmektedir.

Tablo5.1. Kısmi optimizasyon yöntemi ile elde edilen optimizasyon sonuçları

35637323

		OPT_1	OPT_2	OPT_3	OPT_4	OPT_5	OPT_6	OPT_7	OPT_8	Önlenmiş Hata Biti Sayısı
1.BÖLGE	Farklı Bit Sayısı	22059	22077	22039	22085	22057	22053	22055	22041	20
	Aynı Bit Sayısı	7125	7107	7145	7099	7127	7131	7129	7143	
	% Aynı Bit Oranı	%24,41	%24,35	%24,48	%24,32	%24,42	%24,41	%24,42	%24,47	
2.BÖLGE	Farklı Bit Sayısı	24576	24584	24568	24564	24548	24564	24576	24574	28
	Aynı Bit Sayısı	4608	4600	4616	4620	4636	4620	4608	4610	
	% Aynı Bit Oranı	%15,78	%15,76	%15,81	%15,83	%15,88	%15,83	%15,78	%15,79	
3.BÖLGE	Farklı Bit Sayısı	21367	21419	21373	21403	21411	21365	21387	21369	2
	Aynı Bit Sayısı	7817	7765	7811	7781	7773	7819	7797	7815	
	% Aynı Bit Oranı	%26,78	%26,60	%26,76	%26,66	%26,63	%26,79	%26,71	%26,77	
4.BÖLGE	Farklı Bit Sayısı	15091	15139	15061	15099	15167	15175	15121	15229	30
	Aynı Bit Sayısı	14093	14045	14123	14085	14017	14009	14063	13955	
	% Aynı Bit Oranı	%48,29	%48,12	%48,39	%48,26	%48,02	%48,00	%48,18	%47,81	
		144463						144237		
5.BÖLGE	Farklı Bit Sayısı	15880	15820	15850	15866	15942	15854	15782	15822	98
	Aynı Bit Sayısı	13304	13364	13334	13318	13242	13330	13402	13362	
	% Aynı Bit Oranı	%45,58	%45,79	%45,68	%45,63	%45,37	%45,67	%45,92	%45,78	
6.BÖLGE	Farklı Bit Sayısı	15852	15860	15842	15880	15892	15892	15892	15842	10
	Aynı Bit Sayısı	13332	13324	13342	13304	13292	13292	13292	13342	
	% Aynı Bit Oranı	%45,68	%45,65	%45,71	%45,58	%45,54	%45,54	%45,54	%45,71	
7.BÖLGE	Farklı Bit Sayısı	14946	14918	14932	14928	14934	14924	14936	14918	28
	Aynı Bit Sayısı	14238	14266	14252	14256	14250	14260	14248	14266	
	% Aynı Bit Oranı	%48,78	%48,88	%48,83	%48,84	%48,82	%48,86	%48,82	%48,88	
8.BÖLGE	Farklı Bit Sayısı	14692	14696	14682	14698	14720	14710	14690	14720	10
	Aynı Bit Sayısı	14492	14488	14502	14486	14464	14474	14494	14464	
	% Aynı Bit Oranı	%49,65	%49,64	%49,69	%49,63	%49,56	%49,59	%49,66	%49,56	
	% Orjinal Alan	%0,000								Toplam : 226

Şekil 5.17 incelendiğinde tüm işlemin 1dk 47,18sn'de tamamlandığı görülmektedir. Tablo 5.1'de optimizasyon uygulamaları ve sonuçları görülmektedir. Buradan gizleme işlemi sonucu örtü verisi üzerinde oluşan toplam hatalı bit sayısının 144237 bit olduğu görülmektedir. Burada OPT_1 olarak adlandırılan 1. sütunda klasik LSB tekniğinin de kullandığı sırayı içeren birinci optimizasyon verileri görülmektedir. Bu örneklemede LSB klasik kodlama sütunu sekiz farklı optimizasyonun hiçbirisinde minimum hata oranını yakalayamamıştır. Buradan en az hata oranına sahip olamadığı anlaşılmaktadır. Yapılan optimizasyon tarama tekniği ile ilk bölge için OPT_3 en optimum sonucu vermiştir. Çünkü burada 22039 bit hatalı 7145 bit ise aynen korunarak toplam %24,48'lik orijinal olarak korunan bit yüzdesi elde edilmiştir. Bu oran klasik LSB tekniğinde %24,41'de kalmıştır. Yalnızca bu bölgede 20 bitlik ekstra hata engellenmiştir. Benzer şekilde ikinci bölgede OPT_5 en optimum sonucu vererek % 15,88'lik bir başarı sağlamış ve LSB tekniğine göre 28 bit hata önleme iyileşmesi temin edilmiştir. Bu örnek uygulamada tüm bölgelerdeki kazanılan hata önleme bitlerinin toplamı 226 olarak hesaplanmıştır.

Aynı veriler üzerinde gizleme işleminin bu kez lzw sıkıştırma algoritması tabanlı optimizasyon tekniği ile uygulanmasıyla elde edilen sonuçlar Şekil 5.18 ve Tablo 5.2'de görülmektedir.

Şekil 5.18. Lzw sıkıştırması tabanlı kısmi optimizasyon yöntemi ile veri gömme uygulaması

Tablo5.2. Lzw sıkıştırması tabanlı kısmi optimizasyon yöntemi ile elde edilen optimizasyon sonuçları

74514528

		OPT_1	OPT_2	OPT_3	OPT_4	OPT_5	OPT_6	OPT_7	OPT_8	Önlenmiş Hata Biti Sayısı
1.BÖLGE	Farklı Bit Sayısı	5275	5277	5277	5281	5285	5289	5269	5281	6
	Aynı Bit Sayısı	3989	3987	3987	3983	3979	3975	3995	3983	
	% Aynı Bit Oranı	%43,05	%43,03	%43,03	%42,99	%42,95	%43,05	%43,12	%42,99	
2.BÖLGE	Farklı Bit Sayısı	5321	5299	5319	5295	5303	5319	5311	5335	26
	Aynı Bit Sayısı	3943	3965	3945	3969	3961	3945	3953	3929	
	% Aynı Bit Oranı	%42,56	%42,80	%42,58	%42,84	%42,75	%42,58	%42,67	%42,41	
3.BÖLGE	Farklı Bit Sayısı	5889	5909	5893	5911	5875	5903	5937	5913	14
	Aynı Bit Sayısı	3375	3355	3371	3353	3389	3361	3327	3351	
	% Aynı Bit Oranı	%36,43	%36,21	%36,38	%36,19	%36,58	%36,28	%35,91	%36,17	
4.BÖLGE	Farklı Bit Sayısı	5774	5824	5774	5820	5808	5818	5806	5814	0
	Aynı Bit Sayısı	3490	3440	3490	3444	3456	3446	3458	3450	
	% Aynı Bit Oranı	%37,67	%37,13	%37,67	%37,17	%37,30	%37,19	%37,32	%37,24	
		44391						44041		
5.BÖLGE	Farklı Bit Sayısı	5798	5788	5780	5742	5768	5832	5794	5776	56
	Aynı Bit Sayısı	3466	3476	3484	3522	3496	3432	3470	3488	
	% Aynı Bit Oranı	%37,41	%37,52	%37,60	%38,01	%37,73	%37,04	%37,45	%37,65	
6.BÖLGE	Farklı Bit Sayısı	5404	5410	5396	5398	5340	5394	5440	5414	64
	Aynı Bit Sayısı	3860	3854	3868	3866	3924	3870	3824	3850	
	% Aynı Bit Oranı	%41,66	%41,60	%41,75	%41,73	%42,35	%41,77	%41,27	%41,55	
7.BÖLGE	Farklı Bit Sayısı	5422	5312	5418	5346	5334	5314	5392	5334	110
	Aynı Bit Sayısı	3842	3952	3846	3918	3930	3950	3872	3930	
	% Aynı Bit Oranı	%41,47	%42,65	%41,51	%42,29	%42,42	%42,63	%41,79	%42,42	
8.BÖLGE	Farklı Bit Sayısı	5508	5502	5466	5504	5454	5474	5472	5434	74
	Aynı Bit Sayısı	4356	4362	4398	4360	4410	4390	4392	4430	
	% Aynı Bit Oranı	%47,02	%47,08	%47,47	%47,06	%47,60	%47,38	%47,40	%47,81	
	% Orjinal Alan	%0,000								Toplam : 350

Şekil 5.18 incelendiğinde gizlenecek olan verinin %69 oranında sıkıştırılarak boyutunun 9339 bayt(9K)'a indirgendiği ve tüm işlemin yalnızca 36,6sn'de tamamlandığı görülmektedir.

Bu örneklemede LSB klasik kodlama sütunu sekiz farklı optimizasyonun içerisinde sadece 4. bölgede minimum hata oranını yakalayabilmiştir. Buradan en az hata oranına sahip olamadığı anlaşılmaktadır. Yapılan optimizasyon tarama tekniği ile yedinci bölge için OPT_2 en optimum sonucu vermiştir. Çünkü burada 5312 bit hatalı 3952 bit ise aynen korunarak toplam %42,65'lik orijinal olarak korunan bit yüzdesi elde edilmiştir. Bu oran klasik LSB tekniğinde %41,47'de kalmıştır. Yalnızca bu bölgede 110 bitlik ekstra hata engellenmiştir. Benzer şekilde sekizinci bölgede OPT_8 en optimum sonucu vererek % 47,81'lik bir başarı sağlamış ve LSB tekniğine göre 74 bit hata önleme iyileşmesi temin edilmiştir. Bu örnek uygulamada tüm bölgelerdeki kazanılan hata önleme bitlerinin toplamı 350 olarak hesaplanmıştır(Bkz. Tablo 5.2).

Tablo 5.2'de optimizasyon uygulamaları ve sonuçları görülmektedir. Buradan gizleme işlemi sonucu örtü verisi üzerinde oluşan toplam hatalı bit sayısının 44041 bit olduğu görülmektedir. Oysaki önceki uygulamada bu sayının 144237 bit olduğu

bilinmektedir. Bu durumda oluşan toplam hatalı bit sayısında %69'luk bir azalma olduğunu söyleyebiliriz. Yani sıkıştırma oranı ile doğru orantılı olarak toplam hatalı bit sayısı azalmaktadır. Ayrıca bu örneklemede toplam hatalı bit sayısının azalmasının yanı sıra, klasik kodlama olan LSB' ye göre önlenmiş olan hata biti sayısının da daha fazla olduğu görülmektedir.

Deneysel çalışmada, birbirinden farklı boyutlarda, bmp uzantılı görüntü verileri ile yine birbirinden farklı tür(doc, xls, tif, bmp, htm, ppt, txt) ve boyutlarda verilerin, kısmi optimizasyon yöntemi ile gizleme işlemine tabi tutulmasıyla elde edilen sonuçlar Tablo 5.3'te verilmiştir. Burada gizlemede kullanılacak olan örtü verisinin boyutu, içerisine gizlenecek olan verinin boyutu ve türü, işlemin tamamlanması için geçen süre, gizleme işlemi sonucunda elde edilen toplam hatalı bit sayısı ve tekniğin LSB tekniğine göre elde ettiği önlenmiş hata biti sayısı verilmiştir.

Tablo 5.3. Kısmi optimizasyon yöntemi ile gerçekleştirilen deneylere ait sonuçlar

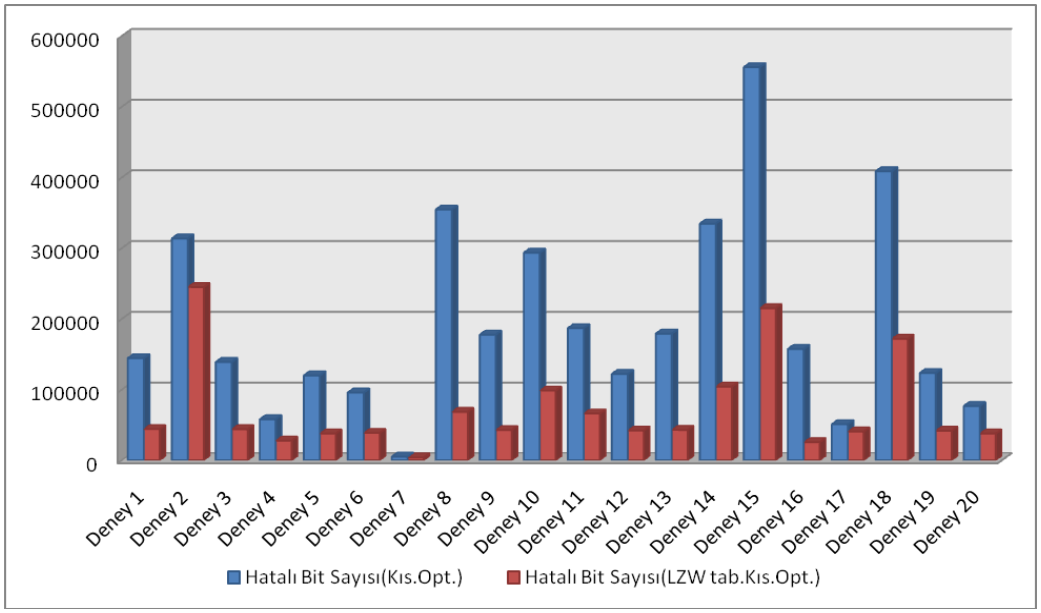
	Gömme Objesi Boyutu(Byte)	Gizlenen Veri Boyutu(Byte)	Gizlenen Veri Türü	İşlem Süresi	Toplam Hatalı Bit Sayısı	Önlenmiş Hata Biti Sayısı
Deney 1	540054	29184	doc	00:01:47	144237	226
Deney 2	748230	78848	doc	00:04:48	313526	852
Deney 3	687286	34664	bmp	00:02:08	138880	950
Deney 4	680678	14676	htm	00:00:54	57876	794
Deney 5	562554	30720	xls	00:01:51	119878	456
Deney 6	786486	24064	xls	00:01:28	95697	232
Deney 7	2494854	1286	tif	00:00:05	5010	120
Deney 8	983094	89148	tif	00:05:31	354527	1348
Deney 9	562554	44825	tif	00:02:41	177552	272
Deney 10	540054	63488	doc	00:03:53	293389	674
Deney 11	525054	46592	doc	00:02:54	186667	516
Deney 12	562554	31232	doc	00:01:54	121987	216
Deney 13	983094	44825	tif	00:02:47	179004	346
Deney 14	680678	84013	htm	00:05:09	334557	1326
Deney 15	2494854	138752	xls	00:08:12	555814	768
Deney 16	540054	42694	bmp	00:02:49	157242	516
Deney 17	151374	12876	bmp	00:00:47	50880	468
Deney 18	983094	101888	ppt	00:07:20	408712	636
Deney 19	687286	31232	doc	00:02:01	123350	486
Deney 20	748230	19350	txt	00:01:15	76687	626

Tablo 5.4'te ise deneysel çalışmada kullanılan aynı veriler, aynı şartlar altında, sıkıştırma tabanlı kısmi optimizasyon işlemine tâbi tutulmasıyla elde edilen sonuçlar gösterilmektedir.

Tablo 5.4. Lzw tabanlı kısmi optimizasyon yöntemi ile gerçekleştirilen deneylere ait sonuçlar

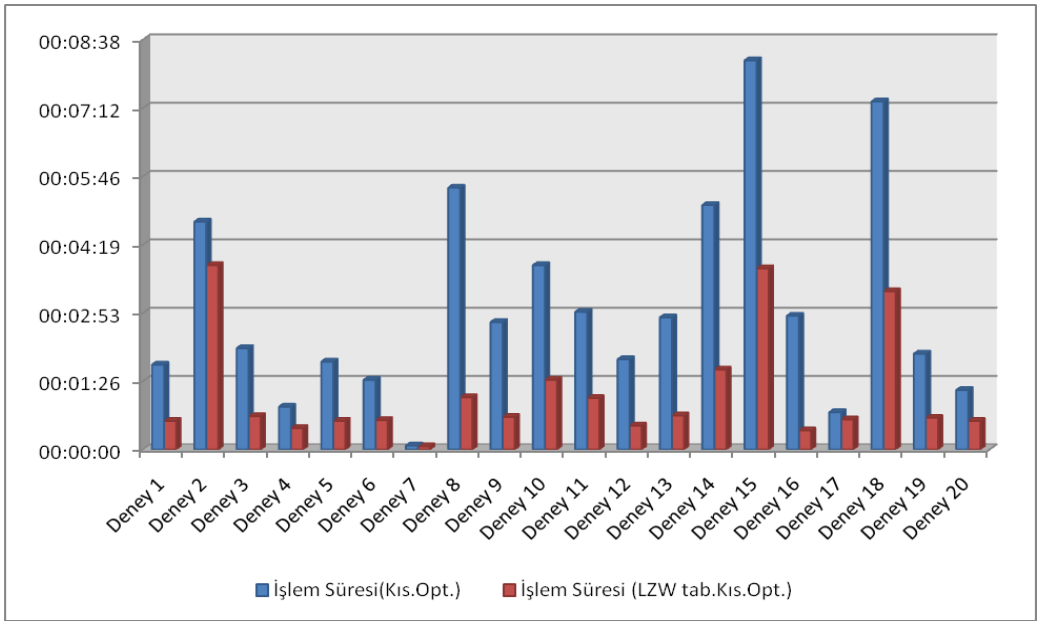
	Gömme Objesi Boyutu(Byte)	Gizlenen Veri Boyutu(Byte)	Sıkıştırma Oranı	Sıkıştırılmış Veri Boyutu(Byte)	İşlem Süresi	Toplam Hatalı Bit Sayısı	Önlenmiş Hata Biti Sayısı
Deney 1	540054	29184	68%	9339	00:00:36	44041	350
Deney 2	748230	78848	22%	61752	00:03:53	244920	1484
Deney 3	687286	34664	68%	11104	00:00:42	43827	486
Deney 4	680678	14676	52%	7050	00:00:27	27720	212
Deney 5	562554	30720	69%	9578	00:00:36	37815	534
Deney 6	786486	24064	59%	9765	00:00:37	38642	558
Deney 7	2494854	1286	15%	1013	00:00:04	3899	170
Deney 8	983094	89148	81%	17261	00:01:06	68411	336
Deney 9	562554	44825	76%	10845	00:00:41	42693	726
Deney 10	540054	63488	63%	23483	00:01:28	98463	562
Deney 11	525054	46592	64%	16689	00:01:05	66057	538
Deney 12	562554	31232	66%	10711	00:00:30	41943	528
Deney 13	983094	44825	76%	10845	00:00:43	42933	614
Deney 14	680678	84013	69%	26156	00:01:41	103966	716
Deney 15	2494854	138752	61%	53954	00:03:49	215014	1410
Deney 16	540054	42694	85%	6278	00:00:24	25486	130
Deney 17	151374	12876	21%	10283	00:00:38	40606	456
Deney 18	983094	101888	58%	43209	00:03:20	171718	842
Deney 19	687286	31232	66%	10644	00:00:40	41818	516
Deney 20	748230	19350	51%	9552	00:00:36	37696	446

Tüm deneylerden elde edilen sonuçlar karşılaştırıldığında sıkıştırma tabanlı optimizasyon uygulamasında, sıkıştırma oranına bağlı olarak hatalı bit sayısının minimize edildiği görülmektedir. Aynı oranda işlem süresinin de kısalması, yönteme getirilen katkılardan bir tanesidir. İşlemlerin tamamlanması için geçen sürenin, deneyin yapıldığı ortam şartlarında, ortalama %58,11 oranında azaldığı görülmüştür.



Şekil 5.19. Deneylerde her iki yönteme ait elde edilen toplam hatalı bit sayısı değerleri

Şekil 5.19'da, her iki yöntem kullanılarak elde edilen sonuçlara göre toplam hata biti sayısının karşılaştırılmasını ifade eden grafik verilmiştir. Grafikte de görüldüğü gibi lzw tabanlı optimizasyon yönteminde hatalı bit sayısı minimize edilerek başarı sağlanmıştır.



Şekil 5.20. Deneylerde her iki yönteme ait elde edilen toplam işlem süresi değerleri

Şekil 5.20'de, gerçekleştirilen yirmi adet deneyde, her iki yöntem kullanılarak elde edilen sonuçlara göre veri gizleme işleminin tamamlanması için geçen sürenin karşılaştırılmasını ifade eden grafik verilmiştir. Grafikte, lzw tabanlı optimizasyon yönteminde işlem süresinin minimize edilmesiyle başarı sağlandığı görülmüştür.

5.3. Sonuç

Bu tez çalışmasında kısmi optimizasyon ile veri gizleme tekniğinin güvenliğini artırmak için gizli verinin algılanabilirliğinin en düşük seviyede tutulması ve bunun daha kısa sürede sağlanması amaçlanmıştır. Bunun için var olan kısmi optimizasyon tekniğine lzw tabanlı bir sıkıştırma algoritması desteği verilmiş, farklı veriler üzerinde deneyler gerçekleştirilerek başarı test edilmiştir.

Deney sonuçlarından elde edilmiş verilere göre, gizli verinin algılanabilirliğinin temelini oluşturan, gizli veriyi içeren görüntü üzerindeki hatalı bit sayısının,

sıkıştırma uygulanmaksızın kullanılan tekniğe göre daha düşük olduğu görülmüştür. Deney sonuçlarından elde edilen grafiklere bakıldığında, geliştirilen sıkıştırma tabanlı yöntem ile veri gizleme işleminin uygulanması için gerekli sürenin minimize edilmesi de elde edilen kazanımlar arasındadır.

BÖLÜM 6. SONUÇ VE ÖNERİLER

Bu tezin amacı resim içerisine veri gizleme amaçlı geliştirilen kısmi optimizasyon tekniğine yeni bir yaklaşım getirerek, bu tekniğin sıkıştırma algoritmasıyla birlikte daha iyi sonuçlar elde etmesini sağlamaktır.

Kısmi optimizasyon yönteminde resim, sekiz farklı optimizasyon bölgesine ayrılarak, her bir bölge sekiz farklı optimizasyon işlemine tabi tutulmaktadır. Bunun amacı optimizasyon kodlaması ile bölgesel kodlama yaparak, her bir bölgede veri gömme sırasını dinamik oluşmasını sağlamaktadır. Bu durumda her bir bölgeye farklı metot uygulanacağından bölgelerarası farklı kodlama teknikleri sebebiyle hatalı kod çözme işlemi en aza indirgenmektedir. Dolayısıyla bu yöntemle, minimum hata oranı için korunan orijinal piksellerin sayısı artırılmakta, ayrıca kendi içerisinde güvenlik sağlanmaktadır. Ancak bu teknik içerisinde bir sıkıştırma algoritması kullanarak korunan orijinal piksellerin sayısı daha da artırılacağı öngörülmektedir. Ayrıca bu yaklaşımın, tekniğin uygulanmasına hız da kazandıracağı aşikârdır.

Veri gizleme uygulamalarında resimde yer alan orijinal piksellerin korunması büyük önem arz etmektedir. Çünkü gizleme amaçlı yapılan bu uygulamanın dışarıdan algılanmaması uygulamanın temel amacını oluşturmaktadır. Bu nedenle kısmi optimizasyon tekniğinin sıkıştırma algoritmasıyla desteklenmesiyle, veri gizleme işleminin daha efektif sonuçlar elde etmesi sağlanmıştır. Bunun için optimizasyon ile gizlenecek olan verinin, kayıpsız ve dinamik sözlük tabanlı bir sıkıştırma tekniği olan LZW sıkıştırma algoritması ile sıkıştırılarak, boyutunun minimize edilmesi sağlanmıştır. Bu sayede optimizasyon ile veri gizleme işlemi sonucu resimde meydana gelen bozulma en aza indirgenmiş ve veri gizleme işleminin daha kısa süre içerisinde gerçekleştirilmesi sağlanmıştır. Bu çalışma, veri gizleme teknikleri üzerine daha önce yapılan çalışmalarda tavsiye edilen bir yöntem üzerine, sıkıştırma tekniğiyle geliştirilerek gerçeklenmiştir.

Gerçekleştirilen deney sonuçlarında işlem sonucu örtü verisinde meydana gelen bozulmanın ve toplam işlem süresinin, gizli verinin sıkıştırma oranı ile aynı oranda azaldığı görülmüştür. Fakat lzw sıkıştırma algoritması sözlük tabanlı bir sıkıştırma yaklaşımı olduğundan, birbirini tekrar eden verilerin yoğunluğu durumunda büyük miktarda sıkıştırma sağlarken, aksi durumda daha az oranda sıkıştırma sağlamaktadır. Sıkıştırma oranı, kısmi optimizasyon uygulamasındaki bozulma ve işlem süresini doğrudan etkilediğinden, kullanılan algoritmanın her veri için yüksek sıkıştırma oranı sağlaması optimizasyon tekniğini daha da kullanışlı hale getirecektir.

KAYNAKLAR

[1] ESİN, E.M., GÜVENOĞLU, E., Resim İçine Yazı Gizlenmesi Amacıyla Kullanılan LSB Ekleme Yönteminin Shuffle Algoritmasıyla İyileştirilmesi, Elektronik Mühendisliği Bölümü, Maltepe Üniversitesi, 2007.

[2] MESUT, A.Ş., MESUT, A., SAKALLI, M.T., Görüntü Steganografide Gizlilik Paylaşım Şemalarının Kullanılması ve Güvenliğe Etkileri, Bilgisayar Mühendisliği Bölümü, Trakya Üniversitesi, Edirne 2006.

[3] GÜREL, H., Sayısal Resim İçerisine Veri Gizleme Uygulamaları, Elektronik Bilgisayar Eğitimi, Kocaeli Üniversitesi, 2006.

[4] AKAR, F., Veri Gizleme ve Şifreleme Tabanlı Bilgi Güvenliği Uygulaması, Doktora Tezi, Elektronik Bilgisayar Eğitimi ABD, Marmara Üniversitesi, İstanbul, 2005.

[5] MESUT, A., Veri Sıkıştırmada Yeni Yöntemler, Bilgisayar Mühendisliği Anabilim Dalı, Trakya Üniversitesi, Edirne, 2006.

[6] CALDWELL, J., 2nd Lt. Steganography, Crosstalk The Journal of Defense Software Engineering, 25-27 (2003).

[7] KUTUCU, H., KAYA, M., DALKINÇ, M., Criptography and Network Security, Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü 1–2, (2002).

[8] ŞAHİN, A., BULUŞ, E., SAKALLI, M.T., 24-bit Renkli Resimler Üzerinde En Önemsiz Bite Ekleme Yöntemini Kullanarak Bilgi Gizleme, Trakya Üniversitesi J Sci, ISSN 1305-6468, 7(1): 17-22, 2006.

[9] ATICI, M.A., Steganografik Yaklaşımların İncelenmesi, Tasarımı ve Geliştirilmesi, Yüksek Lisans Tezi, Bilgisayar Mühendisliği, Gazi Üniversitesi, Ankara 2007.

[10] ANDERSON, R.J., PETITCOLAS, F.A.P., On The Limits Of Steganography, Journal of Selected Areas in Communications, 16(4): 474-481, May 1998, IEEE.

[11] ZIV, J., LEMPEL, A., A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, Vol. 23, No. 3, pp.337-343, May 1977, IEEE.

[12] ARTZ, D., Digital Steganography: Hiding Data within Data, IEEE Internet Computing, Vol. 5, No.3, pp. 75-80, 2001.

[13] MORKEL, T., ELOFF, J.H.P., OLIVIER, M.S., An Overview of Omage Steganography, Information and Computer Security Architecture (ICSA) Research Group Department of Computer Science, University of Pretoria, South Africa.

[14] ŞAHİN, A., BULUŞ, E., SAKALLI, M.T., Gri Seviye Resimler Üzerinde Rasgele LSB Yöntemini ve Sayı Teorisini Kullanarak Bilgi Gizleme ve Steganaliz, Bilgisayar Mühendisliği Bölümü, Trakya Üniversitesi, 2007.

[15] AMIN, M.M., SALLEH, M., İBRAHİM, S., KATMİN, M.R., SHAMSUDDIN, M.Z.I., Information Hiding Using Steganography, Telecommunication Technology, 4th National Conference, Shah Alam, Malaysia, 21- 25, 2003.

[16] ŞAHİN, A., Görüntü Steganografide Kullanılan Yeni Metodlar ve Bu Metodların Güvenilirlikleri, Doktora Tezi, Bigisayar Mühendisliği ABD, Trakya Üniversitesi, Edirne, 2007.

[17] KURTULDU, Ö., İmge Steganografisi İçin Yeni Yöntemler, Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı, Deniz Harp Okulu, İstanbul, 2008.

[18] AKAR, F., BMP Resimler İçin Veri Gizleme Tabanlı Bilgi Güvenliği Uygulamaları, Elektrik Elektronik Mühendisliği, Deniz Harp Okulu, İstanbul.

[19] AKBAL, T., Ses Verilerine Sıkıştırılmış ve Şifrelenmiş Ham Verilerin Gömülmesi, Yüksek Lisans Tezi, Elektronik ve Bilgisayar Eğitimi, Sakarya Üniversitesi, Sakarya, 2008.

[20] JIANG, J., JONES, S., Word Based Dynamic Algorithms for Data Compression, IEEE Proceedings, Vol. 139, No. 6, December 1992.

[21] SALOMON, D., A Concise Introduction to Data Compression, pp. 61-91, 2008.

[22] SALOMON, D., Motta, G., Handbook of Data Compression, Fifth Edititon, 2010.

[23] LIN, C.H., XIE, Y., WOLF, W., LZW-Based Code Compression for VLIW Embedded Systems, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers, 1530-1591/4, 2004 IEEE.

[24] MESUT, A., CARUS, A., Kayıpsız Görüntü Sıkıştırma Yöntemlerinin Karşılaştırılması, II. Mühendislik Bilimleri Genç Araştırmacılar Kongresi, MBGAK, İstanbul, pp.93-100, 2005.

[25] TAO, T., MUKHERJEE, A., LZW Based Compressed Pattern Matching, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Fl.32816 USA.

[26] ABU TALEB, S.A., MUSAFA, H.M.J., KHTOOM, A.M., GHARAYBIH, I.K., J-C., Improving LZW Image Compression, European Journal of Scientific Research, pp.502-509, 2010 ISSN.

[27] HORSPOOL, R.N., Improving LZW, Dept. of Computer Science, University of Victoria, P.O. Box 3055, Victoria, B.C., Canada V8W 3P6.

[28] SELÇUK, A.A., Word-Based Compression In Full Text Retrieval Systems, M. Sc.Thesis, Bilkent University May, 1995.

[29] KNIESER, M.J., WOLFF, F.G., PAPACHRISTOU, C.A., WEYER, D.J., MCINTYRE, D.R., A Technique for High Ratio LZW Compression, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 1530-1591/3, 2003 IEEE.

[30] ZEEH, C., The Lempel Ziv Algorithm, Seminar, Famous Algorithms, January 16, 2003.

[31] ÇETİN, Ö., ÖZCERİT, A.T., BORU, B., Yüksek Gizli-Bilgi Kapasitesine Sahip Yeni Bir Alındısız Video-Steganografi Yöntemi, Teknik Eğitim Fakültesi Bilgisayar Sistemleri Bölümü, Sakarya Üniversitesi, Electrical&Computer Engineering Dept., University of New Mexico.

[32] ÇİVİCİOĞLU, P., ALÇI, M., Güvenli İletişim İçin Veri Gizleme Tekniklerinin Kullanımı, Elektrik Elektronik – Bilgisayar Mühendisliği 10. Ulusal Kongresi, pp. 422-425, Kayseri.

[33] AKAR, F., VAROL, H.S., A New RGB Weighted Encoding Technique for Efficient Information Hiding in Images, Journal of Naval Science and Engineering, Number 2 Volume 2, July 2004.

[34] WELCH, T. A., A Technique for High-Performance Data Compression, IEEE Computer, 17(6), 8-19, 1984.

[35] ZIV, J., LEMPEL, A., A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, IT-23(3), 337-343, 1977.

[36] SAYOOD, K., Introduction to Data Compression, Morgan Kaufman, San Francisco, California, 1996.

[37] http://marknelson.us/1989/10/01/lzw-data-compression/ (Erişim tarihi: Mart 2012)

[38] ŞAHİN, A., BULUŞ, E., SAKALLI, 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).

[39] AKAR, F., VAROL, H.S., A New RGB Weighted Encoding Technique for Efficient Information Hiding in Images, Journal of Naval Science and Engineering, Volume 2, 21–36, 2004.

[40] KATZENBEISSER S., PETITCOLAS F.A.P., Information Hiding Techniques for Steganography and Digital Watermarking, Artech House, INC. 685 Canton Street Norwood, MA 02062, 2000.

ÖZGEÇMİŞ

Esra Ayça GÜZELDERELİ, 21.04.1988 de Kars'ta doğdu. İlk, orta ve lise eğitimini Ordu'da tamamladı. 2005 yılında Ordu Anadolu Meslek Lisesi, Bilgisayar Bölümünden mezun oldu. 2005 yılında başladığı Karadeniz Teknik Üniversitesi Bilgisayar Teknolojisi ve Programlama bölümünü 2007 yılında bitirdi. 2007 yılında Gazi Üniversitesi, Bilgisayar Öğretmenliği bölümüne girdi ve 2010 yılında mezun oldu. 2010 – 2012 yılları arasında Sakarya Üniversitesi, Elektronik-Bilgisayar Eğitimi bölümünde yüksek lisans eğitimini tamamladı.