**T.C.**

**KIRIKKALE ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

**RESİM DOSYALARI İÇİN STEGANOGRAFİK VERİ GİZLEME**

**PROJE-1**

**Hazırlayan**

**Muhammed Oğuzhan BOZKURT**

**Danışman**

**Dr. Öğr. Üyesi Fahrettin HORASAN**

**Ocak-2021**

**KIRIKKALE**

**RESİM DOSYALARI İÇİN STEGANOGRAFİK VERİ GİZLEME**

**ÖZET**

Steganografinin önemli amacı, gizli bilgiyi, yetkisi olmayan kişilerin görmesi durumunda, bilginin gizlendiği alan dâhilinde bilginin saklandığına dair şüpheleri ortadan kaldırmaktır. Yani steganografinin hedefi, yetkisiz kişilerin saklı bilginin gizlendiğini öğrenmelerine engel olmaktır. Bir steganografi uygulaması, taşıyıcı ortamla ilgili şüpheler oluşturuyorsa, bu durumda yöntem başarılı şekilde sonlandırılamaz. Gizli mesajları ulaştıran kılıflar, dijital ortamda sık-sık kullanılan görüntüler, resimler, sesler veya metinler şeklinde olabilir (Agarwal, 2013:91). Bu çalışmada steganografi teknikleri incelenecek ve bu tekniklerden biri olan LSB (Least Significant Bit – En Zayıf Bit) tekniği seçilecek ve Python dilinde uygulanmaya çalışılacaktır.

**Anahtar Kelimeler:** Steganografi, Veri Gizleme, LSB, Python.

**1.GİRİŞ**

“Yetkisiz erişime karşı bilgiyi koruma görevi her zaman önemli bir konu olmuştur. Eski dönemlerden bu yönde bugün de hala sürdürülen iki ana yöntem bulunmaktadır: kriptografi ve steganografi. Kriptografinin amacı, mesajların içeriğini şifreleyerek gizlemektir. Buna karşılık, steganografi bir mesajın varlığını gizler.”[[1]](#footnote-1)

“Steganografi”, “Yunanca bir kelimedir ve kelime “gizli yazı” anlamına gelir. II. Dünya Savaşı sırasında, ABD hükümeti gizli bilgi aktarma yöntemleriyle mücadeleye büyük önem verdi. Posta ile ilgili bazı kısıtlamalar getirildi. Son yıllarda bilgisayar teknolojisinin gelişimi, bilgisayar steganografisinin gelişimine yeni bir ivme kazandırdı.”[[2]](#footnote-2)

“Steganografinin amacı, gizli bilgiyi, yetkisi olmayan kişilerin görmesi durumunda, bilginin gizlendiği alan dâhilinde bilginin saklandığına dair şüpheleri ortadan kaldırmaktır. Yani steganografinin hedefi, yetkisiz kişilerin saklı bilginin gizlendiğini öğrenmelerine engel olmaktır. Bir steganografi uygulaması, taşıyıcı ortamla ilgili şüpheler oluşturuyorsa, bu durumda yöntem başarılı şekilde sonlandırılamaz. Gizli mesajları ulaştıran kılıflar, dijital ortamda sık-sık kullanılan görüntüler, resimler, sesler veya metinler şeklinde olabilir.” [[3]](#footnote-3)

“Steganografinin farklı bir şekli, filigran yaratma olduğu belirtilmektedir ve ticari ortamda geliştirilerek kullanılmaktadır. Filigran oluşturmada taşıyıcı alan dikkat çekmeyecek şekilde değiştirilerek küçük hacimli bir bilgi saklanılır. Filigran oluşturma esasen, telif hakkı bulunan web sayfaları veya ses dosyaları şeklinde dijital alanların korunmasında kullanılmaktadır. Filigran oluşturma ve steganografi arasındaki tek fark, Filigran oluşturmada kılıf, haberleşme yönündeyse steganografide, saklanan mesaj haberleşme yönündedir.” [[4]](#footnote-4)

Çalışmanın amacı resim dosyalarında insan gözünün ayırt edemeyeceği farklarla veri gizleme uygulaması geliştirmeye çalışmaktır. Bu amaç için Python dili tercih edilmiştir ve ayrıca steganografi yöntemlerinden olan LSB yani “En Zayıf Bit” algoritması seçilmiştir.

Yapacağımız uygulamada kullanıcı üzerine veri gizlenecek resim dosyasını seçecek ve istediği metni giriş alanına yazacaktır. Uygulama, seçilen resmin piksellerinde bulunan renk değerlerine (RGB) metnin karakterlerinin bitlerini yazacaktır. Bu yazma işlemi sadece en zayıf bite yazılacağından dolayı resimde gözle görülür bir değişim olmaz ve böylece istenilen metin gizlenmiş olur.

Araştırmada ilk olarak Steganografi kavramı ve resim dosyaları için kullanılan Steganografi teknikleri incelenecektir.

**2. LİTERATÜR TARAMASI**

“LSB yöntemiyle resim steganografisinin son başarıları üzerinde araştırma yapan iki isim Prashanti ve Sandyarani (2015)’tir. Bu ankette yazarlar, yüksek sağlamlık, yüksek yerleştirme kapasitesi ve gizli bilgilerin tespit edilememesi gibi steganografik sonuçları geliştiren gelişmeleri tartışmaktadır. Bu anketle birlikte iki yeni teknik de önerilmektedir. Birinci teknik, veri veya gizli mesajları kapak görüntüsüne gömmek için kullanılır. İkinci teknikte gizli bir gri ölçekli görüntü başka bir gri ölçekli görüntüye gömülür.”[[5]](#footnote-5)

“Della Baby ve ark. (2015) “Steganografi kullanarak yeni DWT tabanlı Görüntü Koruma yöntemi” önermişlerdir. Çalışmalarında, DWT steganografik tekniği kullanılarak birden fazla RGB görüntüsünün tek RGB görüntüsüne gömüldüğü yeni steganografi tekniği incelenir. Kapak resmi 3 renge ayrılmıştır, yani Kırmızı, Yeşil ve Mavi renk alanı. Bu üç renk alanı gizli bilgileri gizlemek için kullanılır. Bu sistem kullanılarak elde edilen deneysel sonuçlar iyi bir sağlamlığa sahiptir. PSNR ve SSIM indeksinin değeri yazarlar tarafından stego ve orijinal kapak imajlarının kalitesini karşılaştırmak için kullanılmıştır. Önerilen yöntem iyi düzeyde PSNR ve SSIM indeks değerlerine sahiptir. Yazarlar deneysel sonuçlarının mevcut yaklaşımlardan daha iyi olduğunu ve veri sıkıştırması nedeniyle gömme kapasitesinin arttığını bulmuşlardır. Dolayısıyla yaklaşımlarının genel güvenliği yüksektir ve stego imgesindeki algılanabilir değişiklikler daha azdır.”[[6]](#footnote-6)

“Akhtar ve arkadaşları (2014) geleneksel LSB görüntü steganografi tekniğinin geliştirilmiş versiyonunu sunmakta ve uygulamaktadır. Çalışmaları bit ters çevirme yöntemini kullanarak stego görüntünün kalitesini arttırır. Bit ters çevirme tekniklerine iki yaklaşım önerir ve uygularlar. Bu her iki teknik de, taşıyıcı görüntünün piksellerinin LSB'lerinin yalnızca ve yalnızca belirli piksel bitleri modeliyle ortaya çıktıklarında ters çevrildiği bit ters çevirme tekniklerini çözer. Bu, piksellerde daha az değişikliğe yol açan geleneksel LSB yöntemiyle karşılaştırılır. Gizli mesajın doğru bir şekilde alınması için, ters çevrilmiş bitlerin stego görüntüsü içinde bir yere gömülmesi gerekir. Deneysel sonuçlar, stego görüntüsünün PSNR değerinin iyileştiğini göstermektedir; dolayısıyla stego görüntü kalitesi iyileştirilir.”[[7]](#footnote-7)

“Deshmuk ve arkadaşları (2014) ayrıca LSB ikamesine dayanan kenar uyarlamalı steganografiyi sunmaktadır. Uyarlayıcı şema ve taşıyıcı görüntünün iki bitişik pikseli arasındaki fark kullanılarak gizli bilgileri taşıyıcı görüntünün keskin (kenarlar) bölgelerine gömerler. Teknikleri diğer LSB ve Pixel fark tabanlı tekniklerden daha iyi performans gösterir ve stego görüntüsünün kalitesini korur.”[[8]](#footnote-8)

“Yang ve arkadaşları (2009), görüntü steganografisi için yeni bir uyarlanabilir LSB tabanlı yöntem sundu. Daha iyi stego görüntü kalitesi için piksel ayarlama tekniğini kullanır. Bu uyarlanabilir LSB ikamesi, yüksek gizli kapasiteye neden olur. LSB tabanlı görüntü steganografi yöntemi önerilmektedir. Verileri gizlemek için ortak bit kalıbı kullanılır. Mesaja ve desen bitlerine göre piksellerin LSB'leri değiştirilir. Bu yöntem düşük gizli kapasiteye sahiptir.”[[9]](#footnote-9)

**3. STEGANOGRAFİ**

**3.1 Steganografi Kavramı**

Steganografi, “gizli bilgiyi başka bir ortamın içine gizleme sanatı ve bilimidir. Dijital steganografinin temel amacı, iletim sırasında herhangi bir davetsiz misafir tarafından fark edilmeden internet üzerinden gizli veri göndermektir.”[[10]](#footnote-10)

““Steganos” kapak, “grafia” yazı anlamına gelir.”[[11]](#footnote-11) Steganografi, “bir tür bilgiyi diğer bilgilerin kapağında saklayan tekniğin adıdır.”[[12]](#footnote-12) Steganografi, “bazı taraflarda şifreleme yerine diğer taraflara daha az şüphe verdiği için tercih edilmektedir.”[[13]](#footnote-13) “Son yıllarda steganografi tekniklerini araştırmak ve geliştirmek ve bunları ticari olarak uygulamak için uluslararası ilgi artmaktadır.”[[14]](#footnote-14)

“Bir yazı sisteminin geliştirilmesi stenografinin gerçek başlangıcıdır. Eski zamanlarda Sümerler çivi yazısı denilen sembolik bir yazı biçimi yarattılar. Yumuşak kildeki resimleri işaretlemek için derme çatma bir kalem kullanarak yazıları kaydettiler. Bu yazı sistemi Mezopotamya'da MÖ 100'e kadar kullanılmıştır. Son haliyle, telaffuza yardımcı olacak fonogramlar içeriyordu.”[[15]](#footnote-15)

“Suudi Arabistan'da Kral Abdul-Aziz bilim ve yenilik kentinde, 12 yıl önce yazıldığı kabul edilen gizli yazı hakkında bazı eski Arapça orijinal kopyalarını İngilizceye dönüştürmek için bir proje başlatıldı. Bu kompozisyonların bazıları Türkiye ve Almanya'da bulundu.”[[16]](#footnote-16)

Steganografi yöntemleri; LSB, DCT, BPCS olarak üçe ayrılmaktadır. Çalışmamızda LSB yöntemi kullanılmıştır. LSB, “gizli verileri görüntü piksellerinde olduğu gibi kapak veri baytlarının en az önemli bitlerine ileten bir steganografi yöntemidir. Steganografide kullanılan en ünlü, temel ve basit gömme yöntemidir. Yöntemin algılanamazlığı ve sağlamlığının parametrelerini geliştirmek için LSB yöntemine birçok iyileştirme yaklaşımı uygulanmıştır. Dijital steganografiyle ilgilenen herhangi bir araştırmacı için, LSB'yi diğer birçok steganografi yöntemine gömmenin en eski ve temel konsepti olarak inceleyerek başlayın. Temel LSB yönteminde, pikseller sıralı bir biçimde gömmek için kullanılır.”[[17]](#footnote-17) LSB gizlenmesi, “daha sonra açıklanacak olan rastgele bir şekilde de yapılabilir.”[[18]](#footnote-18)

“Verilerin LSB yöntemine nasıl gömüleceğini anlamak için LSB, herhangi bir ikili değerin sağ tarafından son bit olarak bulunur.”[[19]](#footnote-19) “Örneğin, (11110101) bayt ikili değerine sahipsek, ilk LSB (1) ve ilk iki LSB biti (01) 'dir, ayrıca son üç LSB bitinin (101) eşit olduğunu bilmek istiyorsak. Örnekte, LSM gömme için, veri gömme için bayt veya pikselin ikili değerinde yalnızca son biti, son iki biti veya son üç biti kullanmak mümkün olduğunu belirtmemiz gerekir. Bu 28 nedenle, 1 bayt piksel boyutumuz varsa, minimum LSB kapasitesi piksel başına 1 bittir (bpp).”[[20]](#footnote-20)

“LSB'ye uygulanan iyileştirme yöntemlerinden biri, LSB gömme için kapak ortam parçalarının (pikseller) rasgele seçimi için psödondom kodlaması kullanmaktır. Rastgele gömme sözde rasgele sayı üreteci (PRNG) kullanılarak yapılabilir.”[[21]](#footnote-21) “LSB steganografisinde kullanılan özel yalancı benzersiz kod aslında yöntemin anahtarıdır. Yalancı kod anahtarı olmadan mesaj alıcı tarafından çıkarılamaz.”[[22]](#footnote-22) “Rasgeleleştirme yöntemi, kapak ortamındaki rasgele gürültü ile aynı özelliklere sahiptir. Bu nedenle, gömülü veriler üçüncü taraflarca gürültü olarak kabul edilebilir ve bu, sözde kodlama kullanmanın amacıdır.”[[23]](#footnote-23)

Least Significant Bit ya da Türkçe adıyla En Zayıf (Önemsiz) Bit’e ekleme yöntemi en yaygın kullanılan Steganografi yöntemlerinden biridir. Uygulaması ve çalışma biçimi oldukça basittir. Yöntem, resim dosyalarında bulunan pikseller üzerinde değişim yapılarak uygulanır. Her piksel kendi içerisinde 4 değer barındırır. Bunlar; pikselin görünürlüğünü yani opaklığını ifade eden “alpha” değeri ve 3 renk değeridir (RGB). Bu 4 değer 0-255 byte arasındaki sayılarla ifade edilir. Bu yöntem uygulanırken 3 renk değeri kullanılır. Öncelikle kullanıcının gizlemek istediği metnin tüm karakterlerine binary dönüşümü uygulanır. Örneğin “merhaba” metni için;

m – 1101101, e – 1100101, r – 1110010, h – 1101000, a – 1100001, b – 1100010, a – 1100001

dönüşümleri yapılır ve bu dönüşümlerin tamamı bir string değişkeninde toplanır;

metin = “1101101110010111100101101000110000111000101100001”

Daha sonra resmin pikselleri tek tek okunur ve bu piksellerin renk değerlerinin binary dönüşümlerinin en zayıf bitine “metin” stringindeki karakterler yazılır. Renk değerleri (R:110, G:152, B:180) olan piksele bit yazma işlemini gösterelim;

R: 110 – 1101110, G: 152 – 10011000, B: 180 – 10110100

metin[0] – 1, metin[1] – 1, metin[2] – 0

bu değerleri en düşük bitlerin yerine yazdığımızda yeni sonuç;

R: 111 – 1101111, G: 153 – 10011001, B: 180 – 10110100 olur ve aradaki fark gözle görülemez derece azdır. Bu farkı aşağıdaki fotoğraflarda inceleyebiliriz.

Orijinal renk: (R: 110, G: 152, B: 180) Değiştirilmiş renk: (R: 111, G: 153, B: 180)

Görüldüğü üzere arada gözle görülen bir fark yoktur. Bu şekilde bir resmin tüm renk değerlerini değiştirsek bile üzerinde değişiklik yapılıp yapılmadığı algılanamaz ve bu bize yüksek miktarda veri gizleme olanağı sağlar.

Steganografinin kullanım alanları; metin, görüntü ve ses alanlarıdır. Çalışmamızda görüntü steganografi alanını kullanmış bulunmaktayız.

**4. UYGULAMA**

**4.1 LSB ALGORİTMASININ UYGULANMASI**

LSB algoritmasının nasıl uygulandığını anlatmıştık. Burada bu algoritmanın nasıl gerçekleştiğini daha detaylı bir şekilde sözde kod satırlarıyla anlatmaya çalışacağız.

**4.1.1 Verinin Gizlenmesi**

Kullanıcıdan alınan metnin karakterleri 9 biti tamamlayacak şekilde dönüştürülür ve bir string değişkeninde tutulur. Burada 9 bite tamamlamamızın sebebi Türkçe karakterleri de gizlemek istememizdir. Ayrıca bu stringin sonuna bir de daha sonra çözme işlemi yapılırken yardımcı olması amacıyla bir işaret karakteri eklenir. Bizim çalışmamızda bu karakter ‘\_’ karakteri olarak belirlenmiştir.

Metin = “merhaba” ise,

binaryString = “000110110” + “000111001” + “000011110” + “000010110” + “000100011” + “000000011” + “000100010” + “0001100001” + “001011111” olur.

Bir resim dosyası seçilir ve bu dosyanın piksellerinde gezinilir. Her bir pikselin renk değerlerine “binaryString” de bulunan karakterler eklenir. Bu işlem renk değerlerinin en zayıf bitine uygulanır. Bu işlemin gerçekleştirilebilmesi için öncelikle sayı 1 bit sağa sonra da 1 bit sola shift edilir. Böylece sayının en zayıf biti sıfırlanmış olur. Daha sonra da “binaryString” deki sırası gelen karakterle bu sayı toplanır.

Örneğin (0, 0) indisli pikselin renk değeri (R: 95, G: 68, B:152) ise bu sayıların binary tabanındaki karşılıkları sırasıyla: 1011111, 1000100, 10011000’dur.

Bu sayılar belirtildiği gibi 1 bit sağa sonra da 1 bit sola shift edilirse:

1011111 => 101111 => 1011110

1000100 => 100010 => 1000100

10011000=> 1001100 => 10011000 olurlar.

Daha sonra “binaryString” de bulunan ilk üç karakterle bu sayılar sırasıyla toplanırsa:

binaryString[0] = 0, binaryString[1] = 0, binaryString[2] = 0

1011110 + 0 = 1011110

1000100 + 0 = 1000100

10011000 + 0 = 10011000 olurlar. Elde edilen bu yeni değerlerin ondalık tabandaki karşılıkları sırasıyla; 94, 68 ve 152’dir. Yeni değerler piksele yazılır ve bu işlem “binaryString” in tüm karakterleri yazılana kadar devam eder.

**4.1.2 Verinin Çözülmesi**

Veri çözme işlemi, veri gizleme işleminin tam tersi olarak nitelendirilebilir. Öncelikle veri gizlenmiş olan resim dosyası seçilir. Daha sonra bu resmin piksellerinde gezilmeye başlanır. Her pikselin renk değerleri okunur ve bu renk değerlerinin en zayıf biti alınır. Bu işlem renk değerinin “1” sayısıyla “ve (&)” işlemi uygulanması sonucu gerçekleştirilir. Elde edilen değer bir geçici string değişkeninde tutulur. Bu geçici string değişkeninin uzunluğu 9’a eriştiğinde bir karakter okunmuş demektir ve her okumada bu karakterin işaret karakteri olan ‘\_’ karakterine eşit olup olmadığı sorgulanır. Şayet eşitse işlem tamamen sonlandırılır. Değilse geçici değişkeni metin isimli değişkene aktarılır ve geçici değişken sıfırlanarak işlem tekrar edilir. ‘\_’ karakterinin okunması ve işlemin tamamen sonlanması durumunda elde ettiğimiz metin değişkeninin içeriği, resim dosyasında gizlenmiş olan metnin kendisidir.

Örneğin (0, 0) indisli pikselin renk değerleri (R: 94, G: 68, B:152) olsun. Her renk değerini, 1 sayısı ile “ve” işlemine sokarsak;

94 & 1 => 1011110 & 1 = 0

68 & 1 => 1000100 & 1 = 0

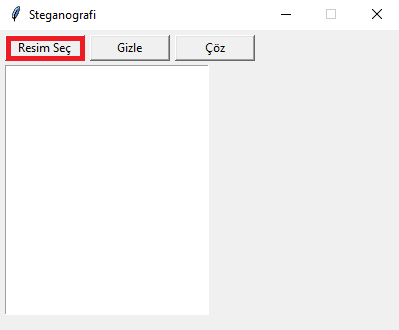
152 & 1 => 10011000 & 1 = 0

gecici = “0” + “0” + “0” olur ve görüldüğü üzere bu değerler daha önce yazdığımız değerlerle örtüşmektedir. Bu da demek oluyor ki veri gizleme işlemimiz başarıyla gerçekleşmiştir.

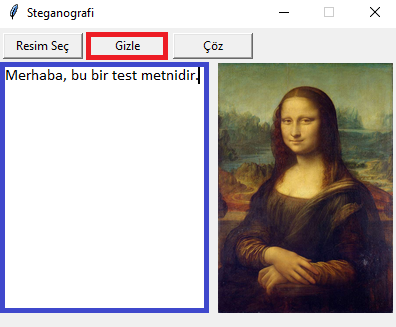
**4.2. ARAYÜZÜN TANITILMASI**

Programın arayüzü aşağıdaki gibidir. Görüldüğü üzere resim dosyasının seçilmesi için “Resim Seç” isimli bir buton, veri gizlemenin yapılması için “Gizle” butonu, gizlenmiş veriyi çözmek için “Çöz” butonu, gizlenecek metnin yazılması için “Entry” alanı bulunmaktadır.

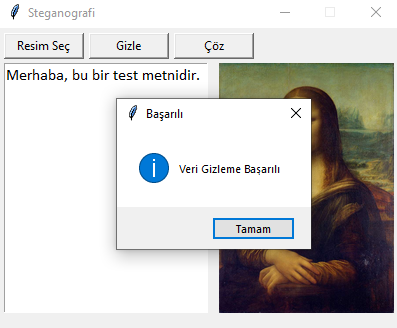
Çalışmaya başlamak için “Resim Seç” butonuna tıklanır.

****

Çalışma yapılacak resim dosyası seçilir ve Entry alanının sağ tarafında resim dosyası gösterilir. Gizlemek için Entry alanına metin girilir.

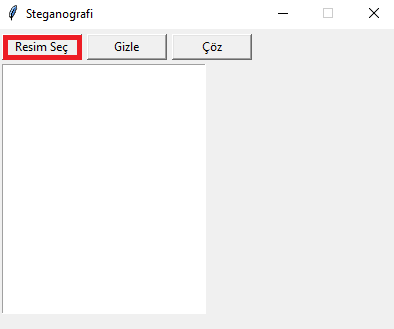
****

Gizleme işleminin yapılabilmesi için “Gizle” butonuna tıklanır.

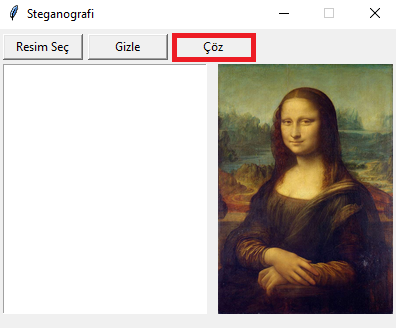
****

Görüldüğü üzere veri gizleme işlemi tamamlandı. Üzerine veri gizlenen resim dosyasına program dizininden erişilebilir. Şimdi çözme işlemi için arayüzü inceleyelim.

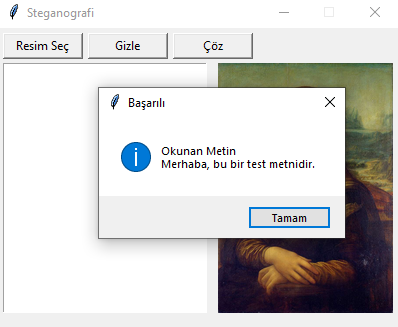
Veri çözme işlemi için resim dosyası seçme işlemi aynı şekilde gerçekleştirilir.

****

Seçilen resim dosyası panelde gösterilir ve çözme işlemi için “Çöz” butonuna tıklanır.

****

Çözülen metin ekranda yeni bir pencerede gösterilir.

****

Görüldüğü üzere veri gizleme ve çözme işlemleri başarıyla gerçekleştirilmiştir.

**5. SONUÇ**

Veri gizleme yöntemlerinden biri olan LSB yöntemini seçtik ve Python dilinde bu algoritmayı Türkçe karakterler de dâhil olacak şekilde kullanıcının istediği metni istediği resim dosyasına gizleyebilmesini sağlayan uygulamayı yazdık. Bu algoritma sayesinde düşük çözünürlüklü resim dosyalarına bile yüksek sayıda karakterden oluşan metinleri, gözlemcilerin algılayamayacağı seviyede renk değişimleri yaparak gizleyebildik.

**KAYNAKÇA**

Agarwal M. (2013). *Text Steganographic Approaches: A Comparison. International Journal of Network Security & Its Applications (IJNSA)*, Vol.5, No.1.

Akhtar N., S. Khan and P. Johri, (2014) *“An Improved Inverted LSB Image Steganography”,* IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Feb. 2014, pp. 749-755.

Baby, D., J. Thomas, G. Augustine, E. George, and N.R. Michael, (2015) *“A Novel DWT based Image Securing method using Steganography”,* International Conference on Information and Communication Technologies (ICICT), Procedia Computer Science, April 2015, pp. 612- 618.

Cheddad, A. J. Condell, K. Curran & Mc. Kevitt. (2010) *Digital image steganography: Survey and analysis of current methods. Signal processing*, 90(3).

Deshmukh P.U. and T.M. Pattewar, (2014) *“A Novel Approach for Edge Adaptive Steganography on LSB Insertion Technique” IEEE International Conference on Information Communication and Embedded Systems (ICICES)*, Feb. 2014, pp. 1-5.

Garg, M. (2011). *A Novel Text Steganography Technique Based on Html Documents. International Journal of Advanced Science and Technology*. Vol. 35.

Johnson, N. F. and S. Jajodia, (1998) *"Exploring Steganography: Seeing the Unseen," IEEE, Computer,* vol. 31, no. 2.

Katzenbeisser S. and F. Petitcolas. (2000) *Information hiding techniques for steganography and digital watermarking. Artech house.*

Kumar K. A., Pabboju S. and Desai N. M. (2014). *Advance Text Steganography Algorithms: An Overview. International Journal of Research and Applications*, 1(1).

Moerland T. (2002). *Steganography and steganalysis,* Leiden Institute of Advanced Computing Science, [www.liacs.nl/home/tmoerl/privtech.pdf](http://www.liacs.nl/home/tmoerl/privtech.pdf).

Petitcolas, F. A., P. R. J. Anderson and M. G. Kuhn, (1999) *"Information Hiding - A Survey," Proceedings of the IEEE*, vol. 87, no. 7.

Pevný, T. (2008). *Kernel methods in steganalysis*. State University of New York at Binghamton,Thomas J. Watson School of Engineering and Applied Science, Department of Computer Science.

Prashanti G. and K. Sandhyarani, (2015) “*A New Approach for Data Hiding with LSB Steganography*”, Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the ComputerSociety of India CSI, Springer 2015, pp. 423-430.

Robert K. (2004) *Steganography and steganalysis*, http://www.krenn.nl/univ/cry/ steg/ article.pdf, January.

Smitha G. L. and E. Baburaj, (2016) *A Survey on Image Steganography Based on Least Significant Bit Matched Revisited (LSBMR) Algorithm,* in International Conference on Emerging Technological Trends (ICETT).

Yang H. , X. Sun and G. Sun, (2009) “*A High-Capacity Image Data Hiding Scheme Using Adaptive LSB Substitution*”, Journal of Radio Engineering, Vol. 18, No. 4, pp. 509-516.

**EKLER**

**Ek 1**

1. **from** tkinter **import** \*
2. **from** PIL **import** Image, ImageTk
3. **from** tkinter **import** filedialog
4. **import** cv2
5. **global** DosyaYolu
7. **def** DosyaSec\_Click():
8. **try**:
9. #Burada Resim Dosyasını Seçiyoruz
10. **global** DosyaYolu
11. DosyaYolu = filedialog.askopenfilename() #Dosya yolunu kullanıcıdan aldırdık
12. Secilen = Image.open(DosyaYolu) #Bu yoldaki resmi açıp bir değişkene atadık
13. Secilen = Secilen.resize((175,250), Image.ANTIALIAS) #Resmin boyutunu değiştirdik
14. Secilen = ImageTk.PhotoImage(Secilen)
15. Resim\_Label = Label(Proje, image=Secilen) #Label nesnesinde bu resmi görüntüledik
16. Resim\_Label.image = Secilen
17. Resim\_Label.place(x=220, y=35, width=175, height=250)
18. **except**:
19. messagebox.showwarning("Dikkat","Resim Dosyası Seçilemedi");
20. **def** VeriGizle\_Click():
21. **try**:
22. #Burada Resim Dosyasına Metni Gizliyoruz
23. **global** DosyaYolu
24. metin = Metin\_Entry.get(1.0, "end-1c"); #Metni Text nesnesinden alıyoruz
25. **if** metin != '':
26. #Metnin tüm karakterlerini 9 bit biçiminde başka bir değişkene atıyoruz
27. binary = ''.join(format(ord(i), '09b') **for** i **in** metin)
28. #Bu değişkenin sonuna '\_' karakterini yazıyoruz bu bizim işaretleyicimiz olucak
29. binary += '001011111'
30. #Resim dosyasını seçilen yoldan okuyoruz
31. img = cv2.imread(DosyaYolu, cv2.IMREAD\_COLOR)
32. height, width, chanel = img.shape #Resmin boyut değerlerini alıyoruz
33. count, x, y = 0, 0, 0
34. #Burada resmin pixellerinde gezeceğiz
35. **while** y < height:
36. **while** x < width:
37. #Döngüler okunan metnin binary dönüşümünün uzunluğu kadar çalışacak
38. **if** count < len(binary):
39. b, g, r = img[y, x] #Pixeldeki renk değerlerini aldık
40. #Renk değerlerini 1 bit sağa sonra 1 bit sola shift ederek son biti yok ettik
41. #Sonra da bu bit yerine binary değişkenindeki sırası gelen biti yazdık
42. r = (r >> 1 << 1) + int(binary[count])
43. count += 1
44. g = (g >> 1 << 1) + int(binary[count])
45. count += 1
46. b = (b >> 1 << 1) + int(binary[count])
47. count += 1
48. img[y, x] = b, g, r #Değiştirilen renk değerlerini resme geri yükledik
49. x += 1
50. **else**: #Döngü bu koşul sağlanmadığında sona erecek
51. y = height
52. x = width
53. y += 1
54. cv2.imwrite("sifreli.png", img) #Yeni resmi kaydettik
55. messagebox.showinfo("Başarılı","Veri Gizleme Başarılı");
56. **else**:
57. messagebox.showwarning("Dikkat","Lütfen Metin Giriniz")
58. **except**:
59. messagebox.showwarning("Dikkat","Lütfen Resim Seçiniz")
61. **def** VeriCoz\_Click():
62. **try**:
63. #Burada Resim Dosyasına Gizlenmiş Metni Çözeceğiz
64. #Resim dosyasını seçilen yoldan okuyoruz
65. **global** DosyaYolu
66. img = cv2.imread(DosyaYolu, cv2.IMREAD\_COLOR)
67. height, width, chanel = img.shape #Resmin boyut değerlerini alıyoruz
68. metin = ''
69. gecici = ''
70. y = 0
71. x = 0
72. #Burada resmin pixellerinde gezeceğiz
73. **while** y < height:
74. **while** x < width:
75. #Döngüler okunan işaretleyici karakter okunana kadar çalışacak
76. b, g, r = img[y, x] #Pixeldeki renk değerlerini aldık
77. #Aldığımız renk değerlerini 1 ile And işlemi yaparak en zayıf bitini elde edeceğiz
78. gecici = gecici + str(r & 1) + str(g & 1) + str(b & 1)
79. **if** len(gecici) == 9:
80. **if** gecici == '001011111': #Burada işaretleyicimizin ('\_') kontrolünü sağlıycaz
81. y = height
82. x = width
83. **else**: #Okuduğumuz karakter işaretleyicimiz değilse aldığımız 9 bitlik sayıyı karakter olarak metne ekliyoruz
84. metin += chr(int(gecici, 2))
85. gecici = ''
86. x += 1
87. y += 1
88. messagebox.showinfo("Başarılı", "Okunan Metin \n" + metin)
89. **except**:
90. messagebox.showwarning("Dikkat","Lütfen Resim Seçiniz")
92. #Burası arayüzümüz
93. Proje = Tk()
94. Proje.title("Steganografi")
95. Proje.geometry('400x300+250+250')
96. Proje.resizable(width=False, height=False)
97. DosyaSec\_Button = Button(Proje, text="Resim Seç", width=10, command=DosyaSec\_Click).place(x=5, y=5)
98. VeriGizle\_Button = Button(Proje, text="Gizle", width=10, command=VeriGizle\_Click).place(x=90, y=5)
99. VeriCoz\_Button = Button(Proje, text="Çöz", width=10, command=VeriCoz\_Click).place(x=175, y=5)
100. Metin\_Entry = Text(Proje, width=25, font="Calibri")
101. Metin\_Entry.place(x=5, y=35, height=250)
102. Proje.mainloop()

1. Kumar K. A., Pabboju S. and Desai N. M. (2014). Advance Text Steganography Algorithms: An Overview. International Journal of Research and Applications, 1(1): p. 31. [↑](#footnote-ref-1)
2. Garg, M. (2011). A Novel Text Steganography Technique Based on Html Documents. International Journal of Advanced Science and Technology. Vol. 35, p. 129. [↑](#footnote-ref-2)
3. Agarwal M. (2013). Text Steganographic Approaches: A Comparison. International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.1. p. 91. [↑](#footnote-ref-3)
4. Agarwal M. (2013). Text Steganographic Approaches: A Comparison. International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.1. p. 93. [↑](#footnote-ref-4)
5. Prashanti G. and K. Sandhyarani, (2015) “A New Approach for Data Hiding with LSB Steganography”, Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the ComputerSociety of India CSI, Springer 2015, pp. 423-430. [↑](#footnote-ref-5)
6. Baby, D., J. Thomas, G. Augustine, E. George, and N.R. Michael, (2015) “ A Novel DWT based Image Securing method using Steganography”, International Conference on Information and Communication Technologies (ICICT), Procedia Computer Science, April 2015, pp. 612- 618. [↑](#footnote-ref-6)
7. Akhtar N., S. Khan and P. Johri, (2014) “An Improved Inverted LSB Image Steganography”, IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Feb. 2014, pp. 749-755. [↑](#footnote-ref-7)
8. Deshmukh P.U. and T.M. Pattewar, (2014) “A Novel Approach for Edge Adaptive Steganography on LSB Insertion Technique” IEEE International Conference on Information Communication and Embedded Systems (ICICES), Feb. 2014, pp. 1-5 [↑](#footnote-ref-8)
9. Yang H. , X. Sun and G. Sun, (2009) “A High-Capacity Image Data Hiding Scheme Using Adaptive LSB Substitution”, Journal of Radio Engineering, Vol. 18, No. 4, pp. 509-516. [↑](#footnote-ref-9)
10. Katzenbeisser S. and F. Petitcolas. (2000) Information hiding techniques for steganography and digital watermarking. Artech house, P. 77. [↑](#footnote-ref-10)
11. Robert K. (2004) Steganography and steganalysis, http://www.krenn.nl/univ/cry/ steg/ article.pdf, January, P. 12. [↑](#footnote-ref-11)
12. Cheddad, A. J. Condell, K. Curran & Mc. Kevitt. (2010) Digital image steganography: Survey and analysis of current methods. Signal processing, 90(3), 727. [↑](#footnote-ref-12)
13. Petitcolas, F. A., P. R. J. Anderson and M. G. Kuhn, (1999) "Information Hiding - A Survey," Proceedings of the IEEE, vol. 87, no. 7, pp. 1062. [↑](#footnote-ref-13)
14. Pevný, T. (2008). Kernel methods in steganalysis. State University of New York at Binghamton,Thomas J. Watson School of Engineering and Applied Science, Department of Computer Science., p. 120. [↑](#footnote-ref-14)
15. Moerland T. (2002). Steganography and steganalysis, Leiden Institute of Advanced Computing Science, [www.liacs.nl/home/tmoerl/privtech.pdf](http://www.liacs.nl/home/tmoerl/privtech.pdf), p. 85. [↑](#footnote-ref-15)
16. Cheddad, A. J. Condell, K. Curran & Mc. Kevitt. (2010) Digital image steganography: Survey and analysis of current methods. Signal processing, 90(3), 727. [↑](#footnote-ref-16)
17. Johnson, N. F. and S. Jajodia, (1998)"Exploring Steganography: Seeing the Unseen," IEEE, Computer, vol. 31, no. 2, pp. 331. [↑](#footnote-ref-17)
18. Smitha G. L. and E. Baburaj, (2016) A Survey on Image Steganography Based on Least Significant Bit Matched Revisited (LSBMR) Algorithm, in International Conference on Emerging Technological Trends (ICETT), p. 47. [↑](#footnote-ref-18)
19. Smitha G. L. and E. Baburaj, (2016) A Survey on Image Steganography Based on Least Significant Bit Matched Revisited (LSBMR) Algorithm, in International Conference on Emerging Technological Trends (ICETT), p. 49. [↑](#footnote-ref-19)
20. Smitha G. L. and E. Baburaj, (2016) A Survey on Image Steganography Based on Least Significant Bit Matched Revisited (LSBMR) Algorithm, in International Conference on Emerging Technological Trends (ICETT), p. 50. [↑](#footnote-ref-20)
21. Smitha G. L. and E. Baburaj, (2016) A Survey on Image Steganography Based on Least Significant Bit Matched Revisited (LSBMR) Algorithm, in International Conference on Emerging Technological Trends (ICETT), p. 57. [↑](#footnote-ref-21)
22. Petitcolas, F. A., P. R. J. Anderson and M. G. Kuhn, (1999) "Information Hiding - A Survey," Proceedings of the IEEE, vol. 87, no. 7, pp. 1062. [↑](#footnote-ref-22)
23. Petitcolas, F. A., P. R. J. Anderson and M. G. Kuhn, (1999) "Information Hiding - A Survey," Proceedings of the IEEE, vol. 87, no. 7, pp. 1065. [↑](#footnote-ref-23)