T.C.
DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ AĞLARDA YENİ BİR ANAHTAR DAĞITIM YÖNTEMİ

ÇAĞATAY AY

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
DOÇ. DR. RESUL KARA

DÜZCE, 2016

T.C.
DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KABLOSUZ AĞLARDA YENİ BİR ANAHTAR DAĞITIM YÖNTEMİ

Çağatay AY tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Resul KARA

Düzce Üniversitesi

Jüri Üyeleri

Doç. Dr. Resul KARA
Düzce Üniversitesi _____

Yrd. Doç. Dr. Okan ERKAYMAZ
Bülent Ecevit Üniversitesi _____

Yrd. Doç. Dr. Mehmet ŞİMŞEK
Düzce Üniversitesi _____

Tez Savunma Tarihi: 19/122016

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

Çağatay Ay

TEŞEKKÜR

Yüksek lisans öğrenimim ve bu tezin hazırlanmasında süresince gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Doç. Dr. Resul KARA'ya en içten dileklerimle teşekkür ederim.

Tez çalışmam boyunca değerli katkılarını esirgemeyen Yrd. Doç. Dr. Esra ŞATIR'a da şükranlarımı sunarım.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen değerli eşim Fethiye Sultan ÖZPEHLİVAN AY, sevgili ailem ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

Çağatay AY

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa No

ÇİZELGE LİSTESİ

Sayfa No

KISALTMALAR

ACB	Associative Coder of Buyanovsky
AES	Advanced Encryption Standard
ASCII	American Standard Code For Information Interchange
AU	Audio Unit
BWT	Burrows-Wheeler Transformation
CALIC	Context Adaptive Lossless Image Compression
CD	Compact Disc
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DMC	Dynamic Markov Compression
DVD	Digital Versatile Disc
FTP	File Transfer Protocol
GZIP	Compressed File Format
HICCUPS	Hidden Communication System for Corrupted Networks
HTML	Hypertext Markup Language
HTTP	Hyper-Text Transfer Protocol
IBM	International Business Machines
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPID	IP Identification
IPV4	IP Version 4
LPC	Linear Predictive Coding
LSB	Least Significant Bit
LZ	Lempel-Ziv
MP3	MPEG-1 Audio Layer III
MTU	Maximum Transmission Unit
NAS	Network Applications and Services
NSA	National Security Agency
NYT	Not Yet Transferred
OFDM	Orthogonal Frequency-Division Multiplexing

PDU	Protocol Data Unit
PPM	Prediction by Partial Matching
RCON	Round Constant
RFC	Request for Comments
RLE	Run-length Encoding
RSA	Rivest-Shamir-Adleman cryptosystem
RST	Reset
SCTP	Stream Control Transmission Protocol
SPN	Substitution-permutation Network
TCP	Transmission Control Protocol
TCPISN	TCP Initial Sequence Number
TTL	Time to Live
UDP	User Datagram Protocol
URG	Urgent
VOIP	Voice Over Internet Protocol
WAV	Waveform Audio File Format
WLAN	Wireless Local Area Network
XOR	Exclusive OR
ZIP	Compressed File Format

ÖZET

KABLOSUZ AĞLARDA YENİ BİR ANAHTAR DAĞITIM YÖNTEMİ

Çağatay AY
Düzce Üniversitesi
Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi
Danışman: Doç. Dr. Resul KARA
Aralık 2016, 84 sayfa

Ağ teknolojilerinin gelişimi ve dijital cihazların artışı multimedya iletimini hızlı ve kolay kılmıştır. Bununla birlikte açık haberleşme kanalları üzerinden yapılan dijital veri iletimi, telif hakkı ihlalleri, dolandırıcılık vb. birçok güvenlik açığını beraberinde getirmiştir. Bu sebepten dolayı güvenli veri iletimi için geliştirilen yöntem ve teknikler oldukça önem kazanmaktadır. Bu tekniklerden biri olan steganografi, gizli iletişim için zararsız görünen bir taşıyıcıya veri eklemesi yapan bilgi gizlemenin alt dallarından biri olarak tanımlanabilir. Veri gizlenirken kullanılan yöntem ve tekniğin sistem dışı kişiler tarafından bilinmesi güvenli veri iletişimini olumsuz yönde etkileyecektir. Steganografinin güvenlik konusunda yetersiz olması, beraberinde şifrelemeyi gündeme getirmektedir. Açık haberleşme kanalları ile yapılmak istenilen gizli iletişimin, çeşitli steganografik metotlar ve şifreleme algoritmaları ile desteklenmesi gerekmektedir. Bu çalışmada, ağ kanalları üzerinden güvenli anahtar dağıtımı için ağ steganografisinden yararlanan bir yöntem geliştirilmiştir. Deneysel sonuçlar, yapılan çalışmanın sağlamlık ve algılanamazlık koşullarında uygulanabilir olduğunu göstermiştir.

Anahtar sözcükler: AES, Ağ Steganografisi, Anahtar Dağıtımı, ICMP, Ping, Uyarlanır Huffman

ABSTRACT

A NEW KEY DELIVERY METHOD IN WIRELESS NETWORK

Çağatay AY
Duzce University
Graduate School of Natural and Applied Sciences, Departmant of Computer Engineering
Master of Science Thesis
Supervisor: Assoc. Prof. Resul KARA
December 2016, 84 pages

The development of networking technologies and the increase of digital devices have made multimedia transmission quick and easy. However, digital data transmission over open communication channels has introduced a number of security implications, such as copyright infringement and fraud. Due to this reason, the methods and techniques developed for secure data transmission gain importance. Steganography, one of these techniques, can be described as one of the subdivisions of information concealment, which makes data linkage to a carrier that seems harmless for occult communication. The methods and techniques of hiding data who are known by non-system users will affect the secure data communication negatively. The inadequacy of the steganography on the security makes encryption important. The secret communication to be made with open communication channels needs to be supported by various steganographic methods and encryption algorithms. In this work, a method has been developed that utilizes network steganography for secure key distribution over network channels. Experimental results have shown that the work performed can be applied to the robustness and imperceptibility conditions.

Keywords: Adaptive Huffman, AES, ICMP, Key Delivery, Network Steganography, Ping

1. GİRİŞ

Ağ teknolojilerinin gelişimi ve dijital cihazların artışı multimedya iletimini hızlı ve kolay kılmıştır. Bununla birlikte açık haberleşme kanalları üzerinden yapılan dijital veri iletimi, telif hakkı ihlalleri, dolandırıcılık vb. birçok güvenlik açığını beraberinde getirmiştir. Bu sebepten dolayı güvenli veri iletimi için geliştirilen yöntem ve teknikler oldukça önem kazanmaktadır [1].

Antik çağlardan buyana gizli haberleşme, gelişen teknoloji ile birlikte uygulama ve yöntem açısından farklılıklar göstermektedir. Gizliliğin oldukça önemli olduğu durumlarda iletilmek istenilen bilgilerin, sistem dışındaki kullanıcıların eline geçmeden hedefe aktarılması amaçlanmaktadır.

Steganografi (steganography) olarak adlandırılan veri gizleme, iletişimin maskelenmesi ile elde edilen gizlilik ve güvenlik durumu olarak tanımlanabilir. Kullanılan yöntemler ve şekil dikkate alındığında veri gizleme, şifrebilim (kriptoloji) ile yakından ilişkilidir. Veri gizleme sanatı olan steganografi ile şifreleme bilimi olarak adlandırılan kriptografi arasındaki en büyük fark, bilgiyi elde eden kişinin elde ettiği veri içerisinde önemli bir bilgi olup olmadığını fark edemiyor olmasıdır. Şifrelemede amaç, aktarılmak istenen mesajın anlaşılmaz hale getirilip ara(gizli) anahtara sahip olmayan kullanıcıların orijinal mesajı elde etmesine engel olmaktır. Çözmesi zor olmasına karşın şifrelenmiş mesaj, ilgi çekici olması sebebiyle gizlenmiş veriyi belli etmektedir [2].

Resim ve ses gürültüsüne bağlı yapılan bazı değişiklikler ile karmaşık şüpheler oluşturularak gizli verinin çeşitli yöntem ve teknikler ile kolayca elde edilememesi sağlanmış olur. Veri gizlenirken kullanılan yöntem ve tekniğin sistem dışı kişiler tarafından bilinmesi güvenli veri iletişimini olumsuz yönde etkileyecektir. Steganografi'nin güvenlik konusunda yeterli olmaması beraberinde şifrelemeyi gündeme getirmektedir.

Farklı iki birim arası haberleşmede verinin güvenli şekilde aktarıldığından emin olmak gerekir. Aktarılmak istenen verinin şifrelenmesi güvenli haberleşmenin temelini oluşturmaktadır. Açık haberleşme kanalları kullanılıyorken sistem dışı kullanıcıların aktarılmak istenen gizli veriye ulaşabileceği, aynı zamanda bu veriyi bozup

değiştirebileceği güvenli haberleşme için önemli bir problem olarak görülebilir [3].

Kriptoloji, kriptografi (şifreleme) ve kriptanaliz (şifre çözme) esas olmak üzere iki bölüme ayrılmaktadır. Gönderilmek istenen mesaj plain text (açık mesaj), bu mesajın şifrelenmiş hali ise cipher text (şifreli mesaj) olarak adlandırılır. Şifreleme, askeri haberleşme başta olmak üzere birçok alanda yaklaşık bin yıldır kullanılmaktadır. Bilgisayar kullanımının yaygınlaşması ve teknolojinin gelişmesiyle birlikte kurumsal ve özel kuruluşlar, iletişimde açık haberleşme kanallarını kullanmaktadırlar. Bu açık kanallar kullanılarak yapılan bilgi aktarımının güvenli ve gizli olabilmesi için şifrelemeye ihtiyaç duyulmaktadır [4].

Şifreleme işleminin gerçekleştirilmesi sırasında kullanılan yöntem ve tekniğin gizliliği, yapılan haberleşmenin güvenliğini belirlemektedir. Yöntem ve tekniğin bilinmesi göz önünde bulundurularak şifreleme işlemi ek bir şifre anahtarı ile gerçekleştirilir. Haberleşmenin güvenliği ve gizliliği, kullanılan yöntem ve teknik ile birlikte şifreleme anahtarı ile doğru orantılıdır.

Açık haberleşme kanalları ile yapılmak istenilen gizli iletişimin, çeşitli steganografik metotlar ve şifreleme algoritmaları ile desteklenmesi gerekmektedir. Gizli verinin steganografi ve şifreleme ile değişime uğraması bilgi gizlemenin temel prensiplerinden algılanamazlık ve sağlamlık açısından oldukça verimli sonuçlar doğururken, kapasite kavramını olumsuz yönde etkileyebilir. Elektronik ortamda iletişimin verimli ve hızlı olabilmesi aktarılmak istenen verinin boyutuyla ters orantılıdır. Hızlı ve etkili bir iletişim beraberinde sıkıştırmayı getirmektedir.

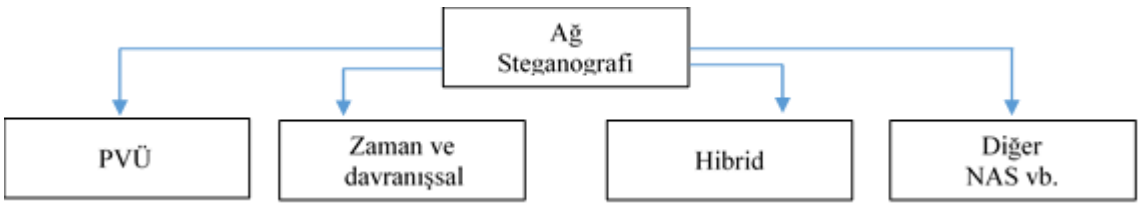
Gerçek zamanlı uygulamalarda bilginin aktarım zamanı oldukça önemlidir. Aktarılmak istenen verinin haberleşme öncesinde sıkıştırılması gerekir. Şifreleme ve steganografi metotları uygulanarak değişime uğrayan orijinal veri, kayıpsız sıkıştırma algoritmaları kullanılarak sıkıştırılıp zaman ve kapasite tasarrufu sağlanmış olur.

Bir veri sıkıştırma işleminde amaç, veri içerisinde bulunan fazlalığı azaltmaktır [5]. Sıkıştırma, kodlama ve kod çözme algoritmalarının kombinasyonudur. Kodlama algoritması verinin sıkıştırılmasını ifade eder. Kod çözme ise sıkıştırılmış verinin orijinal veya orijinale en yakın veriye çıkarılma işlemidir. Bu bağlamda sıkıştırma algoritmaları; kayıplı ve kayıpsız olarak iki sınıfa ayrılmaktadır [6]. Kayıpsız veri sıkıştırma, sıkıştırma işlemi sonucunda çözülen verinin bozulmamış ve bire bir aynı olması istendiği durumlarda kullanılan algoritmadır. Kayıplı veri sıkıştırmada ise çözme işlemi sonucunda

çıkarılan verinin orijinale yakınlığı önemsenmemektedir [7]. Metin tabanlı yapılan uygulamalarda kayıpsız sıkıştırma ve çözme işlemi gerçekleştirilirken orijinal veri korunmalıdır. Dijital resim veya ses bilgisi üzerinde yapılan çalışmalarda çok büyük değişime uğramayan, orijinal veriye yakın dönüşüm yeterli görülmektedir.
Bilgisayar tabanlı steganografik metotlar, gizli kanalı yapılandırmak için genellikle video, ses, resim ve metin gibi dijital medyalardan yararlanır. Son zamanlarda ağ protokolleri steganografik iletişim için yaygın olarak kullanılmaktadır. İçerisinde uygun ağ paketleri bulunan ağ protokolleri, steganografik iletişim için oldukça uygun bir ortam sağlamaktadır [8].

Ağ steganografisi gizli veriyi saklamada kullanıcıların normal veri aktarımlarından yararlanan, üçüncü parti şahıs uygulamalar tarafından algılanması oldukça zor olan yeni bir steganografik eğilimdir. Ağ tabanlı steganografik metotlar veri sızdırma aracı olarak kullanılabileceğinden güvenlik tehditi olarak görülebilir. Bu sebepten ötürü bilgi izleme adımlarını tanıma, karşı önlem alabilmek için oldukça önemlidir.

Ağ steganografi metotları, intra-protocol (protokol-içi) veya inter-protocol (protokol-arası) olarak iki başlıkta sınıflandırılmaktadır. Eğer gizli iletişim, gizli veri taşıyıcısı olarak tek bir ağ protokolü kullanıyorsa bu protokol-içi ağ steganografisi, bir veya daha fazla protokole ihtiyaç duyuyorsa protokol-arası ağ seganografisi olarak adlandırılır. Protokol-içi ağ steganografik metotları 3 alt başlıkta sınıflandırılmaktadır (Şekil 1.1).



Şekil 1.1 Ağ tabanlı steganografik metot sınıflandırması.

PDU (Protokol Veri Ünitesi) düzenleyen steganografik metodlar, ağ protokol başlığı, payload yük alanı veya her ikisini de düzenleyen metotlardır. HICCUPS (Hidden Communication System for Corrupted Networks), damgalama algoritmaları, konuşma kodeği tabanlı steganografik algoritmalar IP, TCP, ve UDP başlık alanlarını düzenleyen metotlara örnektir.

Zaman ve davranış bazlı steganografik metotlar, paket sırasını, paketler arası gecikmeyi veya kasıtlı yapılan paket kayıplarını baz almaktadır. Bu metotlar kullanılarak yapılan çalışmalarda algılanamazlık oldukça yüksek iken aktarılan steganografik veri

kapasitesi oldukça düşüktür. Bu tarz metotlar kullanmak aktarımın kalitesini etkileyebilir.

Hybrid steganografik metotlar, paketin yapısıyla birlikte zamanlama ve sıra düzenlemesi yapar. Bu tarz metotların kullanımında algılanamazlık yüksek, bant genişliği kullanımı zaman tabanlı metotlara göre düşüktür fakat bu durum aktarım kalitesini etkileyebilir [9].

Bunların dışında ağ uygulama ve servislerini kullanan çeşitli metotlar da mevcuttur. NAS tabanlı metotlar ağ uygulamaları ve servislerinin karakteristik özelliklerinden yararlanarak harici veriyi yerleştirmektedir.

Ağ tabanlı steganografik metotlar bant genişliği, sağlamlık ve algılanamazlık gibi üç temel kavram üzerinden değerlendirilmektedir. Steganografik bant genişliği, zaman içerisinde aktarılan harici veri miktarı ile ölçülebilir. Steganografik sağlamlık, gürültü ve taşıyıcı manipülasyonu nedeniyle harici veri üzerinde oluşabilecek değişimin direnci olarak adlandırılabilir. Son olarak algılanamazlık, aykırı algılama durumlarına karşı koyabilmek olarak nitelendirilir [8].

1.1 LİTERATÜR TARAMASI

Harici veriyi ağ protokollerinde yerleştirmek için olası uygun ortamlar arasından oldukça dikkat çekeni IPV4'ün IPID(IP Identification) alanıdır. IPID'in RFC dokümanlarında tanımlı olmasına rağmen nasıl uygulanacağına dair talimatlar belirtilmemiştir. Her geliştirici IPID'i farklı şekilde uygulayabilir fakat RFC'de belirtilen standartlara uymak gerekir. IP protokolünde IPID'in kullanılma amacı IP parçalama süreci sonrasında paketi tekrar oluşturmaktır. IP parçalama, birleştirildiğinde orijinal formu koruyacak şekilde küçük parçalara ayırma işlemidir. Oluşturulan bu küçük paketler birbirlerine bağlanarak aktarım işlemi gerçekleştirilir. Bu paketler maksimum aktarım ünitesi(MTU) tarafından belirtilen boyuttan büyük olamaz. MTU temel ağ bağlantıları tarafından ele alınan maksimum paket boyutu olarak tanımlanmaktadır [10]. Bölünen parçaların alıcı tarafında birleştirilmesinde IPID, parçaları birbiri ile ilişkilendirmekte görevlidir.

PDU tabanlı metotlar genel olarak daha az karmaşık ve yüksek steganografik bant genişliği kullanmaktadır. Gizli kanal geliştirilirken genellikle paket genişliği istismar edilir [11]. Geliştirilen bir metotta IP paketinde bulunan TTL(Time to Live) alanı kullanılarak harici verinin paket içerisine yerleştirilmesi gerçekleştirilmiştir [12]. Sunulan bu metot, gönderilen son TTL'i tekrarlayarak '0' veya '1' kodlaması yapar. Bir diğer

sunulan steganografik metotta Etherleak güvenlik açığı kullanılarak ethernet veri paketine harici veri yerleştirme işlemi yapılmıştır [13] [14].Yapılan bir diğer çalışmada gizli kanal yapısı oluşturulurken IPID ve TCPISN alanları kullanılmıştır [15]. Ahsan ve ark., yapmış oldukları çalışmada gizli mesajın aktarılmasında IPID'in yüksek 8 bitini kullanmayı önermiştir [16]. Hintz, çalışan TCP protokolünü analiz etmiş ve TCP paketi içerisindeki URG alanını '0'a eşitleyerek gizli mesajı aktarmayı sağlamıştır [17]. Benzer bir çalışmada TCP paketinde yer alan RST alanı ve ICMP paketinde bulunan payload alanı gizli mesajın aktarılmasında kullanılmıştır [18]. Kablosuz iletişimde sıra kontrol ve parça kontrol gizli mesajın aktarılmasında kullanılmış diğer alanlara örnek olabilir [19].

Zaman ve davranış tabanlı metotlarda harici veriyi yerleştirme işleminde ağ protokolünün davranışsal mekanizmasından yararlanılır. TCP gibi güvenilir ağ protokollerinde tekrar aktarım mekanizmaları, harici veriyi göndermek için kullanılmıştır [20] [19].Geliştirilen bu metotta bilerek onaylanmayan paketlerin tekrar gönderimi sırasında harici veri pakete yerleştirilir. Bunların dışında ağ paketinin alım zamanı, harici veriyi zararsız taşıyıcıya yerleştirmek için kullanmıştır [21]. Bu metotta önceden belirlenmiş zaman aralığında paketin ulaşıp ulaşmama durumu '1' veya '0' olarak kodlanır. Bir diğer örnekte SCTP'nin multi-stream ve multi-homing özellikleri harici verinin aktarılmasında kullanılmıştır [22] [23]. Ji ve ark., yapmış oldukları çalışmada farklı paket uzunlukları kullanılarak gizli mesajın aktarılmasını sağlayan bir metot geliştirmişlerdir [11]. Cabuk ve ark., ardışıl gönderilen iki paketin arasındaki zaman gecikmesinin düzenlenmesiyle gizli bitleri kodlayan bir metot geliştirmişlerdir [21]. Yapılan benzer bir çalışmada paketin dağıtım aralığını temel alan bir ON/OFF gizli zamanlama kanalı sunulmuştur [24].

Son olarak NAS tabanlı steganografik metotlar ağ uygulamaları ve servislerinin karakteristik özelliklerinden yararlanmaktadır. Bu alanda geliştirilmiş olan bir metotta, Skype ve VoIP üzerinden harici veriyi yerleştirmek için şifrelenmiş sessiz sinyal paketi kullanılmıştır [25]. Yapılan bir çalışmada ardışıl sıralanmış HTTP parametreleri kullanılarak gizli mesajı kodlayan bir metot geliştirilmiştir [26]. Gizli mesajın aktarılmasında FTP protokolü kullanan, yapmış olduğu çalışmada FTP protokolü içerisinde yer alan boş komutlardan yararlanmıştır [27]. Szczypiorsky ve ark., yapmış oldukları çalışmada IEEE 802.11 OFDM sembolleri içerisine veri gizlemeyi başarmış ve WLAN içerisindeki HICCUPS'ın performansı üzerine tartışmışlardır [28] [29].

Yapısı ile karşılaştırma yapıldığında gizli ağ kanallarının fark edilmesi oldukça güçtür. Bu konuda geliştirilmiş birkaç metot ve çalışma mevcuttur. Tumoian ve ark., TCPISN

gizli kanalını algılamak için tarafsız ağ kullanmayı ileri sürmüştür [30]. Murdock ve ark.,TCP gizli kanalı algılamak için istatistiksel bir metot tasarlamıştır [31]. Bunların dışında geliştirilen bazı algılama metotları belirli gizli kanallarda başarılı sonuçlar elde etmiştir.

1.2 AMAÇ, YÖNTEM VE TEKNİK

Yapılan bu çalışmada, kablosuz ağlarda anahtar dağıtımı problemi ile başa çıkabilmek için ağ steganografisinden yararlanmak amaçlanmaktadır. Bu amaç doğrultusunda anahtar dağıtımı için ICMP protokolünü kullanan ping komutundan yararlanılmıştır. Ping, ICMP protokolü içerisinde yer alan, birbirleri arasında haberleşecek iki birim arasındaki durumu belirleyen ve taraflara iletişim hakkında geri bildirim yapmayı sağlayan bir yazılımdır. Kısaca ping, hedef birimin durum ve tepki süresi hakkında bilgi sahibi olmamızı sağlayan program parçacığı olarak adlandırılabilir. Yapılan bu çalışmada, ping paketinin ağı test etmek için gönderildiği ilk seferde harici olarak aktarmak istediğimiz anahtar verimiz bu paket içerisine yerleştirilecek ve alıcıya gönderilecektir. Böylelikle ekstra yük gerektirmeden anahtar teslimi yapılmış olacaktır.

Anahtar bilgisiyle makul format ve boyutta hazırlanan içerik, ping paketi içerisinde bulunan data alanına yerleştirilir. Ping komutunun çalıştırılmasıyla hazırlanan harici veri ping paketi üzerinden alıcıya gönderilir ve bu şekilde kablosuz ağlarda anahtar dağıtımı gerçekleştirilmiş olur.

İlk aşamada onaltılık anahtar, veri paketi içinde aranır ve verinin paket içerisindeki konum bilgisi ile bir dizi oluşturulur. Bir sonraki aşamada bu dizi etkili ve hızlı bir iletişim sağlanması için sıkıştırılır. Sıkıştırma işlemi yine son yıllarda oldukça popüler olan, birbirini tekrar eden veriler üzerinde sıkıştırma oranının yüksek olduğu Uyarlanır Huffman algoritması ile gerçekleştirilir. Sıkıştırılan dizi, veri güvenliği ve şifreleme standardı olarak kullanılan AES ile şifrelenir. Sıkıştırılıp şifrelenen veri, ping paketi içerisinde bulunan data alanına gizlenip ilgili alıcıya anahtar aktarımı yapılır. Alıcı tarafında paket ayrıştırılıp ters algoritma adımları uygulanarak anahtar veri elde edilir. Teknik açıdan önerilen metot, gizli veriyi saklamak için ağ paketinin içeriğini düzenleyen protokol-içi ağ steganografisi alanı kapsamındadır.

1.3 TEZ ORGANİZASYONU

Yapılan tez çalışması 8 temel bölümden oluşmaktadır.

Bölüm 1: Tez çalışmasına konu olan problemin tanımı, çalışmanın amacı, problemin çözümü ile ilgili kullanılan kavramların tanıtımı ve problem ile ilgili literatürde yapılan benzer çalışmaların özetleri sunulmuştur. Aynı zamanda yapılan tez çalışmasında önerilen metot hakkında amaçlar belirtilerek kullanılan yöntem ve teknikler kısaca açıklanmıştır.

Bölüm 2,3,4,5,6: Tez çalışmasında kullanılan yöntemler detaylı bir şekilde açıklanmıştır. Bilgi gizlemenin temeli olan steganografi, açık haberleşmede veri güvenliği sağlayan kriptoloji, etkili ve hızlı bir iletişimin sağlanması için kullanılan sıkıştırma metotları detaylı bir şekilde açıklanmış, kablosuz ağlarda anahtar dağıtımı problemi ile ilgili geliştirilen yöntem sunulmuştur.

Bölüm 7: Geliştirilen uygulama ve yöntem ile ilgili deneysel çalışmalar belirtilmiştir.

Bölüm 8: Tez çalışmasına ait sonuç ve öneriler bu bölümde yer almaktadır.

2. BİLGİ GİZLEME VE STEGANOGRAFİ

Teknolojinin gelişmesiyle son yıllarda elektronik ortam kullanan sistemlerin güvenliği oldukça önemli bir konu olarak ön plana çıkmaktadır. İnternetin yaygınlaşması, mobil kullanım ve bulut teknolojisi ile birlikte veri alışverişi artmıştır. Özellikle sosyal medya kanallarıyla metin, resim, video, ses vb. dosyalar dünyanın her tarafına gerçek zamanlı paylaşılabilmektedir. Anlık iletişimi kolaylaştıran bu haberleşme ağı ciddi güvenlik açıklarını da beraberinde getirmektedir. Birbiri ile haberleşmekte olan iki kullanıcı arasındaki iletişim, sistem dışı istenmeyen bir kullanıcı tarafından dinlenebilir veya düzenlenebilir. Bu durum önemli bir haberleşme açığı olarak değerlendirilmektedir.

Haberleşmede önemli bir tehdit olan bu durumu engellemek için birçok yöntem, teknik ve uygulama geliştirilmiştir. Bunlardan biri şifrelemedir. Şifreleme ile haberleşmede korunması istenen veri, isteğe bağlı şifreleme algoritmaları ve harici bir anahtar ile şifrelenir. Kullanılan yöntemin bilinmesi, şifreleme ile yapılan iletişimi tehlikeye sokabileceğinden güvenli iletişim için bilgi gizleme yöntemlerine başvurulabilmektedir.

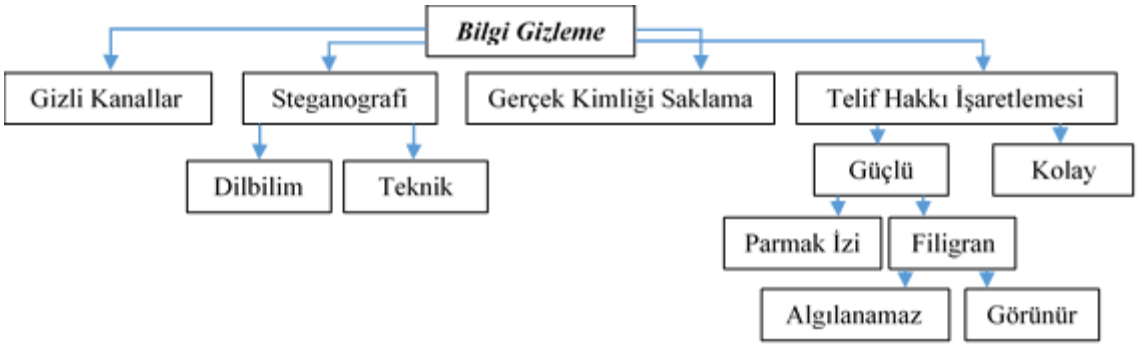
Bilgi gizleme, güvenli haberleşme açısından oldukça önemli bir konudur. Bilgi gizlemenin amacı yapılan haberleşmenin sistem dışı kullanıcılar tarafından algılanamaz olmasını sağlamaktır. Şifrelemede sistem dışı kullanıcılar aktarılan verinin şifrelendiğinden haberdarken, bilgi gizleme yöntemleriyle yapılan haberleşmelerde bu çok mümkün değildir. Zararsız bir taşıyıcı üzerinden aktarılan gizli verinin fark edilmesi oldukça zordur.

2.1 BİLGİ GİZLEME

Bilgi gizleme, iki birim arasında yapılan haberleşmenin sistem dışı kullanıcılar tarafından fark edilmesini engellemek olarak tanımlanabilir. Bilgi gizlemede amaç, haberleşmede kullanılan veriyi saklamaktır. Özellikle askeri ve istihbarat birimleri olmak üzere banka vb. özel kuruluşlar, birimler arası haberleşmenin gizli ve dış kullanıcılar tarafından erişilemiyor olmasını amaçlamaktadırlar. Şifrelenerek aktarılmak istenen verinin, çeşitli yöntemlerle ele geçirilmesiyle, kullanılan şifreleme algoritmasına bağlı olarak zaman

içerisinde çözülmesi mümkündür. Bu sebeple, güvenliğin üst düzey olması beklenen haberleşmelerde çeşitli bilgi gizleme teknikleri kullanılmaktadır.

Haberleşmenin gizli olabilmesi için birçok yöntem geliştirilmiştir. Bu yöntemler Şekil 2.1 de sınıflandırılmıştır [32].



Şekil 2.1 Bilgi gizleme yöntemleri.

Yapılan bu tez çalışmasında bilgi gizleme yöntemlerinin en önemli alt dallarından biri olan steganografi yaklaşımı kullanılmıştır. Steganografi herhangi bir nesnenin içerisine veri gizlemesi olarak adlandırılmaktadır. Dijital ses, resim ve videolar, içerisine veri gizlenebilen zararsız taşıyıcılara örnektir. Yapılan çalışmada, ICMP protokolü kullanan ve aynı zamanda ağın durumunu kontrol eden ping komutu kullanılarak anahtar dağıtımını gerçekleştirilmektedir. Anahtar veri ilk ping paketi içerisine gizlenerek alıcıya gönderilir ve gizli haberleşme sağlanmış olur.

2.2 STEGANOGRAFİ

Steganografi bilgi gizleme yöntemlerinin en önemli alt dallarından biridir. Eski bir bilgi gizleme sanatı olan steganografi, Yunanca steganos; gizli ve grafi; yazım kelimelerin birleşiminden türetilmiş olup *gizlenmiş yazı* olarak adlandırılmaktadır [33]. Bu yaklaşımda içine bilgi gizlenecek zararsız taşıyıcıya örtü verisi, oluşan ortama ise stego-nesnesi adı verilir (Şekil 2.2) [34].

$$Mesaj + Örtü\text{-}Nesne = Stego\text{-}Nesne$$

Şekil 2.2 Stego-Nesne.

Steganografi, tarihi oldukça eskiye uzanan bir veri gizleme sanatıdır. Bilinen en eski steganografik metot Yunan tarihçi Heredot'un eserinde bahsettiği, İran'da bulunan bir casusun saçlarını kazıtıp gizli mesajı kafa derisi üzerine dövme yaptırarak Yunanistan'da

mesajı bekleyen alıcıya iletmesidir. Burada gizli mesaj, saçlarını kazıtmış olan casusun kafa derisi üzerine dövme olarak işlenir. Saçlarının uzamasıyla Yunanistan'a gönderilen casusun bilmesi gereken tek bilgi *"saçımı kazıyın"* olacaktır. Böylelikle gizlenmiş mesaj bekleyen alıcıya iletilmiş olur [35].

Özel kimyasallar ve mürekkepler kullanılarak gizli yazılar elde edilmesi tarihteki ilk steganografik metotlara örnek olmaktadır. Geçmiş zaman içerisinde gizli mesajlar yoluyla haberleşme oldukça yaygındır. Günümüzde dijital resim, video ve ses kanallarının kullanımı gizli verinin aktarımında ideal taşıyıcılar olarak ön plana çıkmaktadır.

Veri gizleme alanında yapılan ilk önemli çalışma, 1954 yılında Muzak şirketinin telif hakkı içeren kod bilgisini ses kayıtlarına yerleştirmesidir [36].

İngiltere Başbakanı Margaret Thatcher, kabine içerisinden dışarıya bilgi sızdıran kişiyi bulmak amacıyla elektronik ortamda bulunan belgelere erişen her kullanıcı için özel tanımlayıcı bilgi eklemiş ve bu yöntem ile bilgi sızdıran kişiyi açığa çıkarmıştır [37].

Sonraki zamanlarda görünmez mürekkep kullanımı, I. ve II. Dünya Savaşlarında kullanılan mors kodları steganografik uygulamalar olarak ön plana çıkmaktadır.

Gelişen teknolojiyle birlikte veri güvenliği amacıyla sıklıkla tercih edilen steganografi, dijital ses, resim, video vb. zararsız taşıyıcılar üzerinde uygulanmaktadır. Benzer bir şekilde sabit disklerde kullanılmayan alanlar, IP paketlerinin ileride kullanılmak üzere ayrılmış bazı bölümleri, HTML dosyaları vb. yapılar gizli verinin saklanması için kullanılabilmektedir.

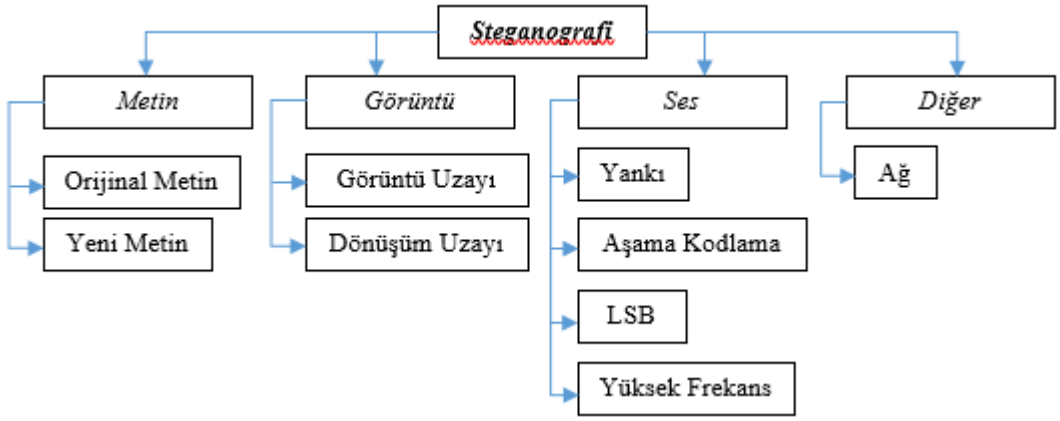
Steganografi, Dilbilim Steganografi ve Teknik Steganografi olmak üzere ikiye ayrılır [38]. Dilbilim Steganografi, taşıyıcı verinin metin formatında olduğu steganografi kolu olarak adlandırılır. Teknik Steganografi ise içerisinde birbirinden farklı birçok konuyu kapsamaktadır. Görünmez mürekkep, gizli yerler, dijital ses, video, resim gibi bilgisayar tabanlı yöntemler Teknik Steganografi kapsamında yer almaktadır.

Kullanım alanları dikkate alındığında steganografi;

- Metin
- Görüntü
- Ses

steganografisi olmak üzere üçe ayrılır.

Yaygın olarak kullanılmakta olan sayısal steganografik yöntemler Şekil 2.3'de sınıflandırılmıştır.



Şekil 2.3 Sayısal Steganografi yöntemlerinin sınıflandırılması [39].

2.2.1 Metin Steganografi

Metin steganografisi adından da anlaşılabileceği üzere bilgi gizlenecek ortamın metin formatında olduğu steganografi yaklaşımıdır. Farklı diller farklı karakteristik özelliklere sahiptir. Dillerin sahip olduğu bu karakteristik özellik, metin steganografisini kullanılan dile bağımlı yapmaktadır [40].

Steganografik metotlar arasında en zor tür metin steganografisidir. Diğer medya türleri ile karşılaştırıldığında metinsel veri daha az fazlalık içerir. Aynı zamanda metinsel veride yer alan anormal görünümler farkındalığı arttırabileceğinden kullanımı oldukça zor olacaktır [41]. Buna karşın daha az hafıza işgali ve basit iletişim sebebiyle tercih edilmektedir [42].

2.2.2 Görüntü Steganografi

Dijital resimler steganografi uygulamalarında yaygın olarak kullanılan aynı zamanda dağıtımı ve erişebilirliği oldukça kolay dosyalardır. Bu nedenle steganografi konusunda yapılan çalışmalar ve geliştirilmekte olan birçok yöntem görüntü steganografisi alanında olmaktadır.

Görüntü tabanlı steganografik metotlarda örtü verisi olarak adlandırılan resim dosyası içerisine gizli bilgi yerleştirilir. Gizlenecek bilgi stego olarak adlandırılır. Gizlenecek bilgi, metin türünde olabileceği gibi görüntü veya bit dizisi içerisine gizlenebilecek herhangi bir şey olabilir. Gizleme işlemi sonucunda elde edilen dosyaya stego resim adı verilir.

Görüntü steganografisi, Görüntü Uzayı ve Dönüşüm Uzayı olmak üzere iki temel başlık altında incelenebilir [43].

Görüntü Uzayı tekniğinde gizleme yapılırken resim dosyasındaki veri doğrudan kullanılır. Resim dosyasında yer alan piksel kümeleri bilginin gizlendiği kısımdır. En Önemsiz Bite Ekleme (LSB), Görüntü Uzayı tekniğinde en sık kullanılan yöntem olarak gösterilebilir.

Dönüşüm Uzayı tekniğinde cover image adı verilen kapak resim üzerindeki değişimlere gizleme işlemi yapılır. Bu teknikte, değişim için DCT ve DFT gibi dönüşümler kullanılmaktadır.

İşaret işlemeye dayalı yöntemler resim üzerinde yapılan değişikliklere karşı da dayanıklıdırlar fakat resmi bozabilirler. Resimde bozulma olup olmayacağı da ancak işlem sonrası anlaşılabilir. Ayrıca, her 64 adetlik bloklara 1 bit gömülebilmesi, saklanabilecek veri miktarını önemli oranda düşürmektedir.

Görüntü dosyaları için bir steganografik sistem Şekil 2.4'de gösterilmektedir. Gönderici bir gizleme fonksiyonu kullanarak bir steganogram yaratır. Gizleme fonksiyonu, verinin saklanacağı taşıyıcı ortam ve gizlenecek veri olmak üzere iki parametreye sahiptir [44].

Şekil 2.4 Stegosistem.

2.2.3 Ses Steganografi

Ses steganografisi, insanın duyma yeteneğindeki geniş aralık nedeniyle oldukça zor bir yöntemdir. İnsan kulağı ses üzerindeki ufak değişikliklere ve ilave gürültülere karşı hassastır. Buna rağmen insan kulağı sesteki bazı bozulmaları ihmal edebilir. Yüksek seslerin yanında yer alan düşük sesler insan kulağı tarafından duyulmamaktadır.

Ses içerisinde veri gizlerken, sesin iletilmesi esnasında maruz kaldığı kodlama ve geri kodlamaya dikkat etmek gerekir. Telefon görüşmeleri 8 kHz, ses CD'lerinde 44.1 kHz,

ses DVD'lerinde 96 kHz ve 192 kHz örnekleme sıklığına sahiptir. Örnekleme sıklığı, sesin frekans spektrumuna bir üst sınır koyarak veri gizlemeyi etkilemektedir.

Ses Steganografi yönteminde, gizli mesajlar dijital sesin içinde gömülüdür. Gizli mesaj ses dosyalarının ikili dizilerinin çok az değiştirilmesiyle yerleştirilir. Bu yöntemde WAV, AU, MP3 gibi ses dosyalarına gizli mesaj yerleştirebilmektedir.

Gizli mesajları saklamak amacıyla birkaç bilgi gizleme metodu bulunmaktadır. Bu metotlar, bilgiyi gürültü içine yerleştiren basit algoritmalardan başlayıp, karmaşık sinyal işlemlerini kullanarak bilgi saklayan çok güçlü algoritmalara doğru çeşitlilik kazanmaktadır. Ses steganografisinde veri gizleme metotları şunlardır:

- Yankı veri gizleme (Echo data hiding)
- Taft yayılması (Spread spectrum)
- En düşük bit kodlama (Low-bit encoding)
- Aşama Kodlama (Phase coding)

Yankı veri gizleme metodunda bilginin gizlenmesi, ses sinyaline yankı eklenmesi ile sağlanmaktadır. Yankını gecikme miktarı, büyüklüğü gibi değerler kullanılarak bilgi gizlemesi yapılır [45].

Taft yayılması metodunda gizleme işlemi, ses sinyalinin kullandığı frekans taftı üzerinde yapılır. Seste gürültü meydana getirmesinden dolayı çok tercih edilen bir yöntem değildir [46].

En düşük bit kodlama yöntemi, görüntü steganografisinde yer alan LSB yöntemi ile aynı şekilde gerçekleştirilir. Ses dosyasında yer alan her 1 bayt'ın en düşük değerlikli son bitine veri gizlemesi yapılır. Yapılan bu değişiklik seste gürültüye neden olabilir [47].

Aşama kodlaması ile bilgi gizleme yönteminde ses dosyası küçük bölümlere ayrılır ve ayrılan her bölüme ait faz gizlenmek istenen veriye ait aşama referansı ile değiştirilir [46].

2.2.4 Diğer Ortamlar

Yapılan steganografik çalışmalarda sıklıkla kullanılan dijital metin, görüntü ve ses dosyalarının dışında, ağ protokolleri, sabit disklerde bulunan kullanılmayan alanlar, HTML dosyaları vb. alanlar gizli veri taşıyıcısı olarak kullanılmaktadır.

Yapılan bu tez çalışmasında bilgi gizleme işlemi, TCP/IP katmanında yer alıp ICMP

protokolünü kullanan ping komutu ile gerçekleştirilmektedir. Ping komutunun kullanmış olduğu ICMP protokolü, içerisinde rahatlıkla bilgi gizlenebilecek birkaç alan bulundurmaktadır. ICMP paketi içerisinde yer alan data alanı gizli veri taşıyıcısı olarak kullanılmaktadır. Data alanı esnek bir yapıda olup varsayılan olarak rasgele 32 baytlık veri bulundurmaktadır. Mevcut verinin modifikasyonu ile gizlenmek istenen bilgi paket içerisine yerleştirilmiş olur.

3. KRİPTOGRAFİ

İnternet ortamına yollanan veri paketleri her ne kadar güvenli ağ içerisinde yer alsa da sistem dışı kullanıcıların bu paketlere erişmesi mümkün olabilmektedir. Son derece gizli bilgilerin yer alabileceği internet ortamı beraberinde güvenlik problemleri ile ilgili kaygıya sebep olmuştur. Gizli bilgiler için güvenli ortam hazırlamadan yapılan haberleşme oldukça risklidir. Başkası tarafından dinlenme, bilgileri değiştirilmesi, kimlik taklidi gibi tehditlerin ortadan kaldırılabilmesi için temel gereksinimlerden biri kriptografidir.

Kriptoloji, Yunanca krypto's (saklı) ve lo'gos (kelime) kelimelerinin birleşmesinden oluşmuştur ve haberleşmede şifre bilimi olarak tanımlanmaktadır [48].

Şifreleme tarihi M.Ö. yıllara kadar uzanmaktadır. M.Ö. Mısırlılar tanrı efsanelerini anlattıkları hiyerografilerin bazı yerlerinde çeşitli semboller kullanmışlardır. Kullandıkları bu semboller ile yazının bazı yerlerini kodladıkları gözlenmiştir. M.Ö. 400 yıllarında Spartalılar *scytale* adı verilen bir yöntem geliştirmişlerdir. Bu yöntemde korunmak istenen bilgiye, silindirik bir sopa üzerine yerleştirilen parşömen ve bu parşömene her bir şerit turuna bir harf gelecek şekilde mesaj yazılmasıyla koruma sağlanmış oluyor. Şeritlerin kaldırılması ile karmaşık yapıda bir mesaj elde edilmiş olur. Mesajın çözülmesi için gerekli olan tek şart şifrelenirken kullanılan ebatlarda silindirik yapı kullanılmasıdır. Farklı çapa sahip silindirik yapılar anlamsız metinlerin oluşmasına sebep olacaktır.

Şifreleme tarihinin en önemli gelişmelerinden biri 2000 yıl kadar önce Julius Ceasar'ın daha sonralarda kendi adıyla anılacak olan bir şifreleme yöntemi kullanmasıdır. Ceasar bu yöntemde mesajda kullanılan her harfi kendisinden sonra gelen üçüncü harf ile değiştirmiştir. Daha sonra Ceasar'ın kullandığı bu yöntem geliştirilerek monoalfabetik yerine koymalı bir yöntem geliştirilmiştir.

II. Dünya savaşında Almanların *Enigma* ve Japonların *Purple* kodlarının kırılması savaşın sonucunu belirleyen önemli faktörler arasında yer almaktadır. Bunun sonucunda kriptografi yeni bir boyut kazanmıştır.

Eskiden sadece askeri çalışmalarda kullanılan şifreleme, internetin yaygınlaşması, mobil teknolojilerin gelişmesi ve bulut tabanlı depolama ile günümüz bilgi çağında kullanımı yaygınlaşmış ve bilgi güvenliği alanında önemli bir araç haline gelmiştir. Askeri kanallar, bankalar, kamu kuruluşları gibi bilgi güvenliğinin önemli olduğu sektörler, kritik öneme sahip bilgilerin iletilmesinde şifreleme kullanmaktadırlar. Şifrelemeye olan ihtiyacın zaman içerisinde artması, şifreleme alanında yapılan çalışmalarda da aynı oranda artışa sebep olmuştur [49].

Kriptografide amaç, birbirleri arasında haberleşmekte olan birimlerin, güvenliği önemsenen veriyi duruma göre şifreleyip ve şifrelenmiş veriyi geri ayrıştırılabilmesidir. Bir bankaya ait hesabınıza erişmek istediğinizde kullanıcı adı ve parola girmeniz gerekmektedir. Aynı zamanda çevrimiçi bir ticaret sitesi üzerinden alışveriş yapmak istediğimizde kredi kartı bilgilerimizi kullanmamız gerekebilir. Çeşitli ağ kanalları üzerinden ilgili sunuculara aktarılan hesap bilgilerimizin sistem dışı kullanıcılar tarafından erişilebilir olmaması öngörülmektedir. Bu durum haberleşmede şifrelemeye ihtiyaç olduğunu gösterir. Ağ ortamına yer alan hesap bilgileriniz, kullanılan platformun güvenlik yapılandırmasına göre şifrelenir ve ağ kanalları üzerinden şifrelenmiş veri paketi şeklinde alıcıya aktarılır. Güvenli haberleşme olarak adlandırılan bu durum kötü niyetli kişilerin güvenli bilgi erişimine engel olacaktır.

Şifrelemede düz bir veriyi karmaşık ve okunmaz hale getirebilmek için çeşitli matematiksel algoritmalar kullanılmaktadır. Bu algoritmalara örnek olarak DES ve günümüzde şifreleme standardı olarak kullanılan AES gösterilebilir [50].

Kullanılan bu algoritmalar düz veriyi ikilik sisteme dönüştürüp belli bölümlere ayırır. Parçalara ayrılmış ikilik veri üzerinde rasgele ve karmaşık matematiksel fonksiyonlar uygulanarak şifreleme gerçekleştirilir [51].

Zaman içerisinde kullanılmakta olan bu algoritmalar kötü niyetli kişiler tarafından çözümlenebilir veya geliştiriciler tarafından paylaşıma açılabilir. Algoritmanın gelişimi ve eksiklerin giderilmesi açısından bu oldukça önemlidir.

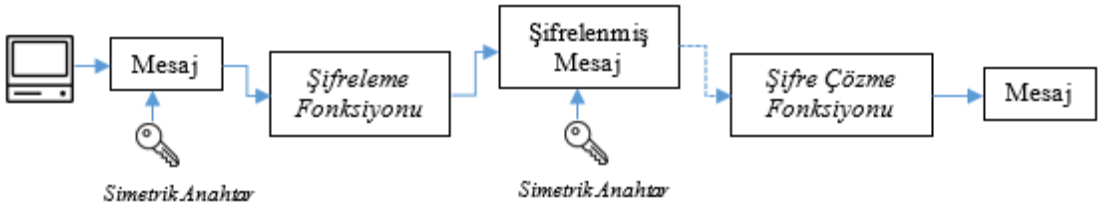
Herkes tarafından bilinen algoritmalar, gizli veya açık bir anahtar kullanarak şifreleme işlemini gerçekleştirmektedirler. Büyük bir çoğunluğu 64 bayta kadar uzanan anahtar veri, içeriğine göre algoritmanın işleyişini değiştirebilmektedir. Anahtar veri üzerine yapılan bir bitlik değişim bile bambaşka bir şifrelenmiş mesajın oluşmasına sebep olacaktır [52].

Yapılan şifreli haberleşmenin herhangi bir değişikliğe uğramadığından emin olmak gerekir. Haberleşme sırasında şifreli mesaj üzerindeki yapılacak 1 bit değişiklik, orijinalinden tamamen farklı bir mesaj elde edilmesine sebep olabilir. Bu durumun önüne geçebilmek için şifreleme işleminde hash fonksiyonu kullanılmaktadır. Hash fonksiyonu orijinal mesaj ile şifreli mesaj arasında herhangi bir değişiklik olup olmadığını kontrol eder. Orijinal mesajdan elde edilen hash, şifrelenmiş mesaj ile alıcıya gönderilir. Alıcı tarafında çözülen mesaj hash fonksiyonuna iletilir. Elde edilen hash kodu ile şifreli mesaj içerisine yerleştirilmiş hash kodu karşılaştırılır. Hash kodunun şifreli mesaj içerisinde gönderilen hash kodundan farklı olması haberleşmenin değişime uğradığını gösterir [53].

Kriptografi, şifreleme işleminde kullanılan anahtarların özelliğine bağlı olarak simetrik ve asimetrik şifreleme olarak iki türe ayrılır.

3.1 SİMETRİK ŞİFRELEME

Bu yöntemde şifrelenmek istenen düz veri, bir anahtar yardımı ile şifreleme algoritmasına alınır. Aynı anahtara sahip alıcı, kullanılan algoritmanın çözme adımlarını uygulayarak şifreyi çözebilmektedir. Simetrik şifrelemede şifreleyici ve şifre çözücü aynı anahtara sahip olmak zorundadır (Şekil 3.1).



Şekil 3.1 Simetrik Şifreleme yapısı.

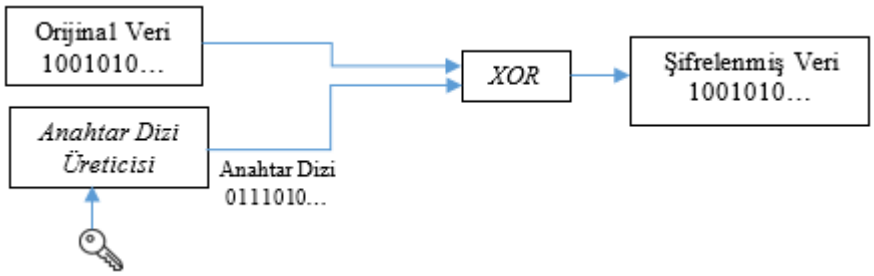
Simetrik şifreleme, kendi içerisinde *Akış Şifreler* ve *Blok Şifreler* olmak üzere iki sınıfa ayrılır.

3.1.1 Akış Şifreler

Akış şifreleme algoritmaları, bit dizisi şeklinde veri alıp yine bit dizisi olacak şekilde çıktı üretir.

Akış şifrelemede ilk adım, gizli anahtar ve anahtar dizi üreteci yardımıyla bir anahtar dizisi oluşturmaktır. Daha sonra orijinal verimizin her bir biti ile anahtar dizimiz *XOR* işlemine tabi tutulur. Yapılan işlemin ardından şifrelenmiş veri elde edilir (Şekil 3.2). Yapılan şifrelemede algoritma güvenliği, anahtar dizisi tarafından üretilen bitlerin rasgele

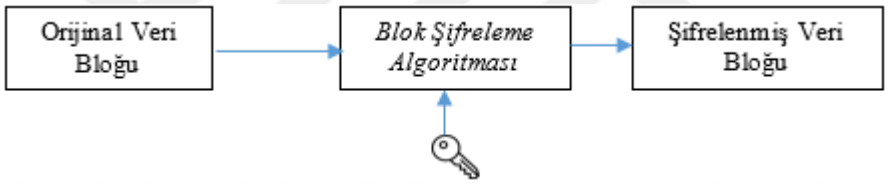
olmasına bağlıdır. Elde edilen anahtar dizisinin kendini tekrarlamaması ve sonraki anahtar bitlerinin öncekiler yardımı ile elde edilememesi anahtar dizi üreticisinin sağlaması gereken önemli özelliklerden biridir.



Şekil 3.2 Akış Şifreleme Yapısı.

3.1.2 Blok Şifreler

Blok şifreleme algoritmaları, orijinal veri olarak bitlerden oluşan veri bloğu kullanır (Şekil 3.3). Günümüzde veri bloğu boyutu olarak 32, 64 veya 128 bit uzunluk kullanılmaktadır.



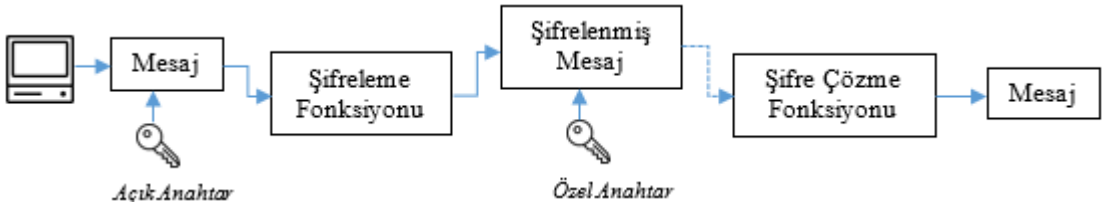
Şekil 3.3 Blok Şifreleme Yapısı.

Blok şifrelerin tasarımı Shannon tarafından ortaya atılan karıştırma ve yayılma tekniklerine dayanır. Karıştırma, şifrenin tek doğrusal olmayan parçası S-kutuları ile sağlanırken yayılma, bit veya bayt tabanlı doğrusal dönüşümlere dayanır. Blok şifreler birden fazla şifreleme işleminin birleşmesi ile oluşturulurlar ve her şifreleme adımı bir döngüdür. Bu döngüler farklı anahtar verisi kullanırlar.

DES ve AES kriptografide önemli yeri olan iki blok şifreleme algoritmasıdır.

3.2 ASİMETRİK ŞİFRELEME

Asimetrik şifreleme yönteminde mesajı şifrelemek için kullanılan anahtar ile şifre çözmek için kullanılan anahtar birbirinden farklıdır. Şifreleme işleminde açık anahtar, şifre çözmede ise özel anahtar kullanılır (Şekil 3.4).



Şekil 3.4 Asitmetrik Şifreleme yapısı.

Simetrik şifreleme yapısında gönderici ve alıcı aynı anahtarı kullanacak olmalarından dolayı göndericinin anahtar bilgisini alıcıya güvenli olarak iletmesi gerekmektedir. Ayrıca simetrik şifrelemede mesajın kim tarafından şifrelendiği açık değildir. Bu iki durum simetrik şifrelemenin dezavantajları arasında yer alır.

Asimetrik şifreleme, simetrik şifrelemenin dezavantajlarına çözüm sunabilmektedir. Asimetrik şifrelemede alıcı özel bir anahtara sahiptir. Gönderici, şifreleme işlemine başlamadan önce anahtar kütüphanesinden sadece sizin özel anahtarınızla açabileceğiniz bir açık anahtar seçer ve düz metin bu açık anahtarla şifrelenir. Bu açık anahtarı çözücü eşi olan özel anahtar ile şifrelenmiş mesaj alıcı tarafında çözülür ve bu şekilde asimetrik şifreleme işlemi gerçekleşmiş olur.

Günümüzde simetrik ve asimetrik şifreleme algoritmaları birlikte kullanılarak hem yüksek derecede güvenlik hem de yüksek hızlı şifreleme gerçekleştirilebilmektedir. Bu gibi sistemlere melez sistem adı verilir. Anahtar şifreleme, anahtar anlaşma ve sayısal imza işlemleri genellikle asimetrik şifrelemeyle, yığın veri işlemleri ve imzasız veri bütünlüğü korumaysa simetrik şifreleme ile gerçekleştirilir.

RSA, Diffie Hellman asimetrik şifreleme tabanlı uygulamalara örnektir. Diffie Hellman, şifreleme anahtarlarının değişiminde kullanılan özel bir yöntemdir. Açık haberleşme kanalları ve güvensiz medyalar üzerinden gizli anahtar paylaşımı yapılmasına olanak sağlar.

Ağ tabanlı uygulamalarda güvenli veri aktarımı sağlanması, sağlıklı iletişim kurulabilmesi adına önemli bir unsurdur. Günümüzde güvenli veri aktarımı için birçok metot geliştirilmiştir. Bunlardan bazıları donanımsal olarak güvenliği sağlarken, önemli

bir kısmı güvenliğini yazılımla sağlamaktadır. Donanımsal olarak firewall cihazları haberleşme birimleri arasındaki güvenliği sağlarken, yazılımsal olarak desteklenen bu sistemler güvenlik açısından yeterli görülmektedir. İletişim halinde olan birimlerin veri aktarımı sırasında şifreleme veya steganografik prosedürlere başvurması güvenliğin yazılımsal olarak sağlanmasına örnektir.

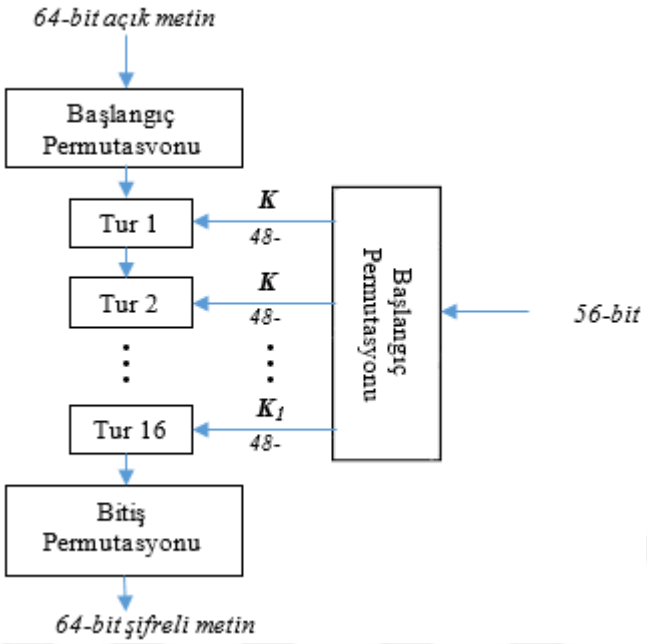
3.3 AES – GELİŞMİŞ ŞİFRELEME STANDARTI

DES (Data Encryption Standart), IBM ve NSA tarafından 1974 yılında geliştirilen, 20 yılı aşkın süre boyunca Veri Şifreleme Standardı olarak yaygın bir şekilde kullanılmış şifreleme algoritmasıdır. Bu süre zarfında algoritma, kriptanalize karşı oldukça başarılı sonuçlar elde etmiştir. Küçük anahtar boyutu kullanıyor olması ve daha büyük boyutlu anahtarlara karşı herhangi bir opsiyonunun bulunmaması zaman içerisinde farklı şifreleme algoritmaları geliştirmeyi ihtiyaç haline getirmiştir.

DES şifreleme standardının en büyük dezavantajı 56-bitlik anahtar kullanıyor olmasıdır. Geliştirildiği dönem dikkate alındığında çok iyi bir şifreleme algoritması olan DES, günümüz teknolojisinde modern bilgisayarların gelişimiyle yapılan saldırılara karşı yetersiz kalmaya başlamıştır. Daha sonraki çalışmalarda DES, Triple-DES olarak geliştirilmiş ve üç kat daha fazla anahtar gücüne sahip olmuştur. 56-bitlik üç adet anahtar kullanarak şifreleme yapan DES, anahtar gücünü 168-bite çıkarırken şifreleme ve dişifreleme işlemi aynı oranda yavaşlamıştır. Kapasitenin artmasından dolayı çevirim işlerinin yavaşlaması DES algoritması için bir dezavantaj oluşturmuştur.

DES, 64-bitlik bloklarda veri şifrelemesi yapan bir blok şifredir. Şifrelenecek olan mesaj parçalara bölünür ve her parça bağımsız olarak şifrelenir. Elde edilen şifrelenmiş mesaj yine aynı şekilde bloklara ayrılarak şifre çözme işlemi yapılır. DES simetrik bir şifreleme algoritmasıdır. Bu durumdan dolayı algoritmanın şifreleme ve şifre çözme işleminde aynı açık anahtar kullanılır.

DES algoritmasına ait yapı Şekil 3.5'de gösterilmiştir.



Şekil 3.5 DES algoritma yapısı.

AES(Advanced Encryption Standart), DES şifre algoritmasının saldırılara karşı dayanıksız olması üzerine kriptografi uzmanları olan Joan Daemen ve Vincent Rijndael tarafından geliştirilmiş, günümüzde hala güvenilirliğini korumakta olan güçlü bir şifreleme algoritmasıdır. 128, 192 ve 256-bitlik anahtar uzunluğu seçeneklerine sahip olan Rijndael algoritması, elektronik ortamda veri güvenliği ve şifreleme standardı olarak kullanılmaktadır.

Günümüzde güvenlik amaçlı kullanılan algoritmalarda ürün boyutunun küçük, aynı zamanda algoritmanın hızlı olması algoritma tercihinde önemli etkenler arasındadır. AES'in minimum sayıda bellek kullanması ve yeterli hızda olması diğer algoritmalar arasında bir adım önde olmasını sağlamaktadır [54].

3.3.1 Şifreleme (Encryption)

Rijndael algoritması, anahtar değerlerine göre farklı sayıda döngüsel işlem yapmaktadır (Çizelge 3.1). Yapılan her döngü sonunda anahtar yenilenir ve bir sonraki döngüde kullanılmak üzere veriye uygulanır. Kullanılan anahtar hem şifreleme hem de şifreli metni çözerken kullanılır. 128-bit uzunluğunda olan veri (4x4)'lük matrislere bölünür. Her satır ve sütun 32 bitlik veriye sahiptir. Oluşturulan bu matrise "durum" matrisi denir ve algoritmanın ilk adımını oluşturur. AES şifreleme algoritmasında kullanılacak olan anahtar uzunluğu doğru orantılı olarak güvenliği arttırırken aynı zamanda işlem sayısını

ve bellek kullanımını da arttırmaktadır.

Çizelge 3.1 Anahtar değerine göre döngüsel işlem sayısı.

Veri	Döngü Sayısı
AES-128	10
AES-192	12
AES-256	14

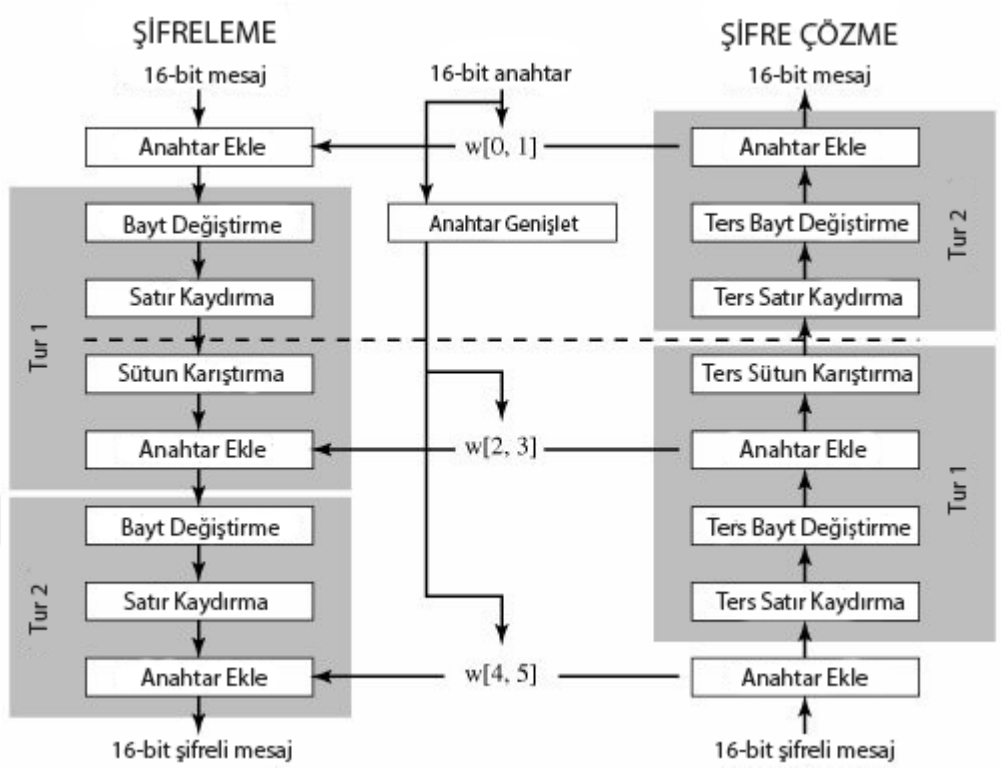
Şifrelemede kullanılacak olan anahtar aynı şekilde durum matrisine dönüştürülür. Veriye ait durum matrisi ile anahtara ait durum matrisinin toplanmasıyla şifrelemenin ilk adımı gerçekleştirilmiş olur.

Algoritmayı güçlü yapan özelliklerden biri de döngü kullanıyor olmasıdır. Algoritma adımları sırasıyla;

- Bayt değiştirme (Sub Bytes),
- Satır kaydırma (Shift Rows),
- Sütun karıştırma (Mix Columns),
- Anahtar ekleme (Add Round Key) işlemlerinden oluşur ve bu işlemler döngünün her adımında tekrarlanmaktadır.

Döngü sayısı başta belirtildiği gibi anahtar uzunluğuna bağlı olarak değişmektedir. Döngünün her adımında üretilen döngü anahtarı bir sonraki döngüde giriş anahtarı olarak kullanılır. Algoritmanın son döngüsünde sütun karıştırma işlemi yapılmaz ve anahtar toplama işlemiyle şifrelenmiş blok elde edilmiş olur. Şifrelenmiş verinin çözümü aynı işlemlerin tersi ile elde edilebilir.

AES algoritmasına ait *Şifreleme* ve *Şifre Çözme* adımları ayrıntılı bir şekilde Şekil 3.6'da gösterilmektedir.



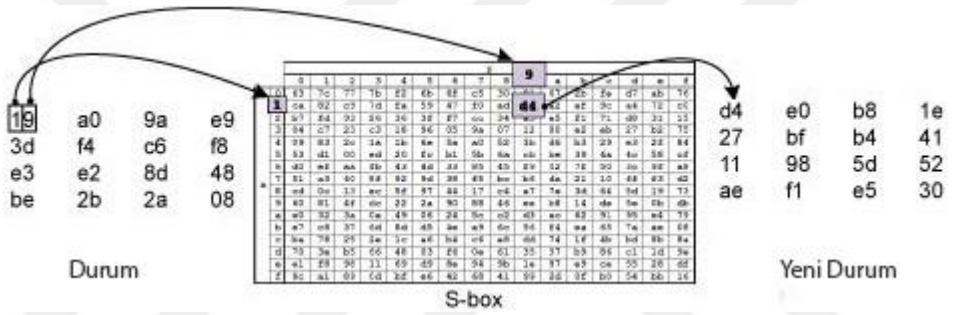
Şekil 3.6 AES Algoritması.

3.3.1.1 *Bayt Değiştirme (SubBytes)*

Döngünün ilk adımı olan bayt değiştirme işlemi başlangıçta oluşturulan veri durum matrisinin eleman değişikliğine uğramasıyla gerçekleşir. Değişiklikler önceden belirlenmiş S-Box (Şekil 3.7)'a göre yapılır. S-Box 16x16 boyutunda bir matris olup göstergeler onaltılık taban elemanları ile yapılır. Şekil – 3.7'e göre durum matrisinin (1,1) indeks numaralı onaltılık 19 değeri 16x16'lık S-Box üzerinde (1,9) indeksinde bulunan onaltılık d4 verisi ile güncellenmiştir. Aynı işlemler durum matrisinde bulunan tüm indeksler için geçerlidir. İşlemler sonucunda yeni durum matrisi elde edilmiş olur.

16x16 boyutlarında bir S-Box ve Bayt Değiştirme Şekil 3.7'de gösterilmiştir.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16



Şekil 3.7 Bayt Değiştirme.

3.3.1.2 *Satır Kaydırma (ShiftRows)*

Satır kaydırma işlemi, bayt değiştirme sonucu oluşan yeni durum matrisi üzerinde yapılır. Bu işlemde matrisin ilk satırı aynı kalırken sırayla ikinci satır 1 bayt, üçüncü satır 2 bayt ve son satır 3 bayt sola kaydırılır. İşlem sonucu yeni bir durum matrisi elde edilmiş olur (Şekil 3.8).

$$\begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix} \quad \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix}$$

Şekil 3.8 Satır Kaydırma.

3.3.1.3 *Sütun Karıştırma(MixColumns)*

Sütun kaydırma işleminde satır kaydırma sonrası oluşan yeni durum matrisi her bir sütun birbirinden bağımsız bir şekilde y(x)={03} x^3+ {01} x^2+ {01}x+ {02} denklemiyle matris çarpımına tabi tutulur (Şekil 3.9). Bu işlemin ardından daha önceki adımlarda olduğu gibi yeni bir durum matrisi elde etmiş oluyoruz.

Şekil 3.9'da Sütün Değiştirme aşamasına ait örnek gösterilmiştir.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} D4 \\ BF \\ 5D \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ E5 \end{bmatrix} \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix}$$

$$\begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix} \Rightarrow \begin{bmatrix} 04 & E0 & 48 & 28 \\ 66 & CB & F8 & 06 \\ 81 & 19 & D3 & 26 \\ E5 & 9A & 7A & 4C \end{bmatrix}$$

Şekil 3.9 Sütun Karıştırma.

3.3.1.4 *Anahtar Ekleme (AddRoundKey)*

AES algoritmasında anahtar üretme, şifreleme işlemi öncesinde yapılmaktadır. Her döngüde farklı bir anahtar oluşturulur. Oluşturulan anahtarlar bir önceki turda oluşturulan anahtarların kullanılmasıyla elde edilir. Anahtar üretme işlemi bir takım sıralı işlem sonucunda olmaktadır. Elimizde bulunan anahtar matrisinin 4. sütununda bulunan 4 baytlık veri 1 bayt üste ötelenir. Öteleme işleminin ardından S-Box kullanılarak bayt değiştirme yapılır ve yeni bir sütun elde edilmiş olur. Elde edilen bu sütun RCON (Şekil 3.10) vektör matrisinin 1. Sütunu ile XOR işlemine tabi tutulur. Oluşan yeni sütun ile 2. sütun tekrar XOR'lama yapılır. Bu işlemler tur sayısı kadar devam eder. Öteleme, bayt değiştirme ve RCON vektörü ile toplama "T-İşlemi" olarak adlandırılmaktadır.

Rcon Constants (Base 16)			
Round	Constant(Rcon)	Round	Constant(Rcon)
1	01 00 00 00	6	20 00 00 00
2	02 00 00 00	7	40 00 00 00
3	04 00 00 00	8	80 00 00 00
4	08 00 00 00	9	1B 00 00 00
5	10 00 00 00	10	36 00 00 00

Şekil 3.10 Byte Değiştirme.

Başlangıçta anahtar üretim bloğu tarafından üretilen 128 bitlik anahtar, döngü sonunda 128 bitlik durum matrisi ile toplanır.

3.3.2 Şifre Çözme (Decryption)

Rijndael algoritmasında şifreli metni çözmek için uygulanan adımlar şifreleme için kullanılan adımların benzeri olup yapılan işlemlerin tersi uygulanması sonucu şifre çözme işlemi gerçekleşir. Şifrelemede kullanılan *Bayt Değiştirme, Satır Kaydırma, Sütun Karıştırma* dönüşümleri ters işlemleri ile değiştirilmelidir. *Anahtar Ekleme* verinin

anahtar ile XOR işlemine sokulması anlamına geldiğinden yapılacak dönüşümün tersi yine kendisi olacaktır. Çözme işlemi sırasıyla;

- Ters Bayt Değiştirme (InSubBytes),
- Ters Satır Kaydırma (InShiftRows),
- Ters Sütun Karıştırma (InMixColumns),
- Ters Anahtar Ekleme (AddRoundKey) işlemlerinden oluşur ve bu işlemler döngünün her adımında tekrarlanmaktadır.

3.3.2.1 *Ters Bayt Değiştirme (InSubBytes)*

Ters bayt değiştirme işleminde bir öncekine benzer şekilde S-Box kullanılmaktadır. Fakat bu S-Box bir öncekinden farklıdır. Bir öncekinde (1,9) indisi D4 onaltılık veriye karşılık gelirken ters bayt işlemi için kullanılan S-Box'da (D,4) indisine 19 onaltılık verisi karşılık gelmektedir.

3.3.2.2 *Ters Satır Kaydırma (InShiftRows)*

Ters satır kaydırma işlemi satır kaydırma işlemin aksine sol taraf yerine sağ tarafa yapılır ve yeni bir durum matrisi elde edilmiş olur.

3.3.2.3 *InMixColumns (Ters Sütun Karıştırma)*

Ters sütun karıştırma işlemi şifrelemede yapılan işlem ile bire bir aynı olup çarpım için kullanılan fonksiyon farklıdır. Durum matrisinin her sütunu y(x)={0B} x^3+ {0D} x^2+ {09}x+ {0E} polinomuyla çarpılır [55].

3.3.3 Güvenlik

Şifreleme algoritmalarının en önemli parçası doğrusal olmayan S kutularıdır. Doğrusal yapıya sahip algoritmalar kolayca kırılabilmektedir. DES şifreleme algoritması ortaya atıldığı zaman bünyesinde yer alan S kutularının bir arka kapıya sahip olabileceği öne sürülmüştür.1990'lı yıllarda Biham ve Shamir, DES algoritmasında kullanılan S kutuları için bir tehdit olabilecek diferansiyel kriptanalizi keşfetmiştir. DES algoritması geliştirildiği zaman IBM geliştiricileri diferansiyel kriptanalizin farkındalardı. Bu bilgiyi gizli tutarak yaklaşık 20 yıl boyunca algoritma güvenliğini korumuştur [56].

DES algoritmasının en büyük zaafı sahip olduğu anahtar uzayı genişliğidir. 2^{56} güçlü bir şifreleme algoritması için oldukça düşük bir anahtar uzayı boyutudur. Bu durum birçok

anahtar arama cihazının geliştirilmesine olanak sağlamıştır. 1998 yılında RSA laboratuvarında geliştirilen DES-*Challenge-II* 56 saat içerisinde bir DES anahtarını bulmuştur.

Anahtar arama saldırıları dışında DES algoritmasına karşı Doğrusal ve Diferansiyel Kriptanaliz olmak üzere iki saldırı tipi mevcuttur.

DES algoritması üzerinde yapılan kriptanaliz, kriptografik çalışmaları da beraberinde getirmiştir. Bu çalışmalardan biri olan AES algoritmasında DES algoritmasından farklı olarak SPN mimarisi kullanılmıştır. DES algoritmasında kullanılan Feistel mimarisinde her döngüde verinin yarısı işlenirken SPN mimarisinde verinin tamamı işlenmektedir ve böylelikle her döngüde veri daha etkin bir biçimde işlenebilmektedir.

AES, SPN mimarisine sahip ve sonlu cisimde ters alma işlemi ile elde edilen bir S kutusuna sahiptir. AES'in sahip olduğu *Sütun Değiştirme* dönüşümü az sayıda S kutusu ile ilişkili doğrusal ve diferansiyel saldırıları mümkün kılmamaktadır. Bu durum geniş deneme stratejisi (wide trail strategy) olarak adlandırılmaktadır [57].

Uzun yıllar boyunca yapılan doğrusal ve diferansiyel kriptanaliz denemeleri durma noktasına gelmiştir. Bu durumdan dolayı saldırıların genelleştirilmesi fikri ortaya çıkmıştır. Vaudenay tarafından ele alınan istatistiksel kriptanaliz fikri Junod tarafından ileri götürülmüştür [58] [59].

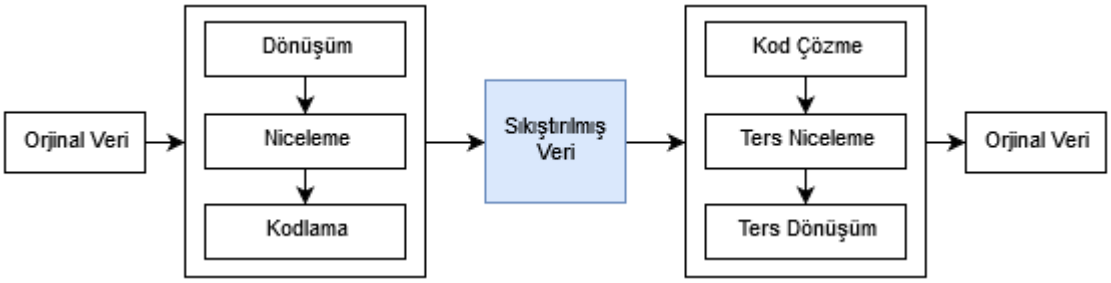
Geliştirilmekte olan saldırı yöntemleri bazı akış şifreler üzerinde olumlu sonuç vermeleri, aynı zamanda kriptografi camiası tarafından oldukça ilgi görmelerine rağmen blok şifreler ve özellikle AES'e karşı bir ilerleme gösterememiştir. Blok şifrelerden ortaya çıkan sistemlerin günümüz hesaplama gücü ile erişilemez olması AES'i güvenli bir veri şifreleme standardı yapmıştır [60].

4. VERİ SIKIŞTIRMA TEKNOLOJİLERİ

Sıkıştırma, bilginin fiziksel olarak boyutunu azaltma işlemi olarak tanımlanabilir. Bir sıkıştırma işleminin, aynı ortam üzerinde daha fazla bilgi depolamak (disk alanı kullanımını azaltmak), ağdaki iletim süresini ve bant genişliğini azaltmak ve dijital bir görüntüyü yeniden kullanmak gibi birçok amacı vardır [61]. Son zamanlarda disk kapasitelerindeki hızlı artış sıkıştırma uygulamalarının kullanımını azaltsa da hızlı haberleşme ve etkin bant genişliği kullanımı dikkate alındığında sıkıştırma, günümüz iletişim teknolojilerinde önemli bir konuma sahiptir. Disklerde yer alan birçok dijital ses, video ve resim dosyası çeşitli yöntemlerle sıkıştırılarak depolanmaktadır.

Veri sıkıştırma yöntemleri kayıplı ve kayıpsız sıkıştırma yöntemleri olmak üzere ikiye ayrılmaktadır. Kayıpsız veri sıkıştırma yönteminde sıkıştırma işlemi sonrası yapılacak olan geri çözme işleminde elde edilen verinin, orijinal veri ile birebir aynı alması gerekir. Veri bütünlüğün korunması ve orijinal verideki bir bit kaybın büyük değişimlere yol açabileceği durumlarda kayıpsız sıkıştırma yöntemleri tercih edilmektedir. Metin bazlı sıkıştırma yöntemlerinde genellikle kayıpsız sıkıştırma metotları tercih edilmektedir. Kayıplı veri sıkıştırma yöntemlerinde sıkıştırma sonrası elde edilen verinin, orijinal veriye yakın bir temsili olması beklenir. Ufak çaplı kayıpların önemsenmediği gerçek zamanlı medya akışlarında, görsel ve işitsel dijital materyallerde sıklıkla tercih edilmektedir. Sıkıştırma oranları dikkate alındığında kayıplı sıkıştırma yöntemleri kayıpsız sıkıştırma yöntemlerine göre bir adım ön plana çıkmaktadır. Kayıpsız sıkıştırma yöntemlerinde 2:1 oranından 8:1 oranına kadar sıkıştırma elde edilebilirken, kayıpsız sıkıştırma metotlarında bu oran 100:1'den 200:1'e kadar çıkmaktadır. Ek olarak, orijinal veri üzerinde fazladan tolere edilebilecek her kayıp, sıkıştırma oranını pozitif yönde etkileyebilecektir [7].

Kayıplı ve kayıpsız bir sıkıştırma işleminin temel aşamaları Şekil 4.1'de gösterilmektedir [62].



Şekil 4.1 Sıkıştırma işlemi temel aşamaları.

Veri sıkıştırma işleminde amaç veri içerisinde yer alan fazlalıklardan arınmaktır [5]. Diskler ve teypler gibi donanımsal bileşenler üzerinde saklanan ya da ağ kanalları üzerinden iletilen veriler önemli ölçüde artıklıklar içermektedir. Veri sıkıştırma ile bu artıklıklar kodlanarak veri yoğunluğu arttırılabilmektedir. Karakter dağılımı, karakter tekrarı, çok kullanılan sözcükler ve konumsal artıklık olmak üzere 4 farklı artıklık türü mevcuttur [63].

4.1 ARTIKLIK TÜRLERİ

4.1.1 Karakter Dağılımı

Bir karakter katarı göz önünde bulundurulduğunda, bazı karakterler diğerlerine göre daha sık kullanılabilir. ASCII karakterlerden oluşan bir belge düşünüldüğünde ASCII tabloda yer alan karakterlerin 4/3'ü kullanılmayabilir. Bunun sonucunda her 8 bitlik paketin 2 bitinden tasarruf edilmiş olacaktır. Bir envanter kaydında sayısal değerlerin kullanımı oldukça yaygındır. Bu şekilde bir metin dizisi, dosya ve belge içerisinde yer alabilecek karakterlerin dağılımı, her karakter için gereken ortalama bit sayısını etkileyecektir.

4.1.2 Karakter Tekrarı

Eğer bir veri bloğu tek bir karakterin tekrarlarından oluşuyor ise bu veri bloğu, olduğundan daha yoğun bir şekilde kodlanabilir. Bazı dosyalarda kullanılmayan alanlar oldukça fazla miktarda yer alabilir. Bir envanter kaydında, alfabetik alanlarda yer alan boşluklar, sayısal alanda yer alan sıfırlar ve kullanılmayan alanlarda yer alan null değerlere oldukça sık rastlanır. Bunların yanı sıra grafik tabanlı görüntüler de çoğunlukla homojen boşluklardan oluşmaktadır.

4.1.3 Çok Kullanılan Sözcükler

Bir doküman içerisinde yer alan bazı sözcükler doküman içerinde yer alan diğer sözcüklere oranla daha fazla sıklıkla kullanılabilirler. Bu sözcüklerin frekans değerlerinin yüksek olması daha az bit sayısı ile temsil edebilmesi anlamına gelmektedir. Örneğin ingilizce hazırlanan bir dokümanda ZE karakter çifti GC karakter çiftine oranla daha fazla yer alabilir. Bu durumda ZE gibi frekans değeri yüksek karakter çiftlerini daha düşük sayıda bitlerle kodlayıp bit kullanımını minimize edebiliriz.

4.1.4 Konumsal Artıklık

Eğer belli karakterler belli veri blokları içerisinde tahmin edilebilir yerlerde bulunuyorlarsa kısmi olarak artıklık taşıdıkları anlamına gelir. Dikey bir çizgi içeren bir resimde her tarama işleminde çizgi aynı konumda olacağından kodlama daha az bit ile gerçekleşebilir.

Bazı durumlarda içerik, dört artıklık türünü de içeriyor olabilir.

4.2 SIKIŞTIRMA ALGORİTMALARI

4.2.1 Kayıpsız Sıkıştırma Algoritmaları

Adından da anlaşılabileceği üzere kayıpsız sıkıştırma yöntemlerinde sıkıştırılmış veriden elde edilen verinin, orijinal veri ile birebir aynı olması amaçlanır. Orijinal veya yeniden oluşturulmuş veri arasındaki herhangi bir farklılığa tolerans gösterilemeyecek uygulamalarda genellikle kayıpsız sıkıştırma algoritmaları tercih edilir. Metin tabanlı verilerin sıkıştırılmasında kayıpsız sıkıştırma teknikleri kullanılmaktadır. Herhangi bir banka kaydı veya önemli yazışmalarda sıkıştırmadan kaynaklanan en ufak değişiklik bile bir felaket ile sonuçlanabilir.

Kayıpsız veri sıkıştırma yöntemleri birçok uygulamada kullanılmaktadır. Günümüzde en yaygın olarak kullanan uygulamalar ZIP ve Unix sistemlerde GZIP'tir. Aynı zamanda bazı görüntü formatları hem kayıplı hem de kayıpsız sıkıştırma yöntemlerini bir arada kullanabilmektedir.

Kayıpsız sıkıştırma algoritmaları genellikle iki aşamada gerçekleşir. Birinci aşamada girdi verisine ait bir istatistiksel model oluşturulur. İkinci aşama ise bu istatistiksel model kullanılarak girdi verisi kodlanır. Bu şamada veri içerisindeki tekrarlar veya olasılıklar önemli rol oynamaktadır.

Huffman ve aritmetik kodlama, kayıpsız sıkıştırma algoritmalarında sıklıkla kullanılan algoritmalardır. Aritmetik kodlama bilgi entropisinde verilen belirli istatistiksel model için uygun olabilen en iyi sıkıştırma oranlarına ulaşabilmekte, Huffman kodlama ise basit ve hızlı bir sıkıştırma olmasına rağmen bire yakın sembol olasılıkları olan modellerde kötü sonuçlar elde edebilmektedir [62].

İstatistiksel model oluşturmak için statik ve uyarlanabilir olmak üzere iki ana model bulunmaktadır. Statik modelde önce veri analiz edilir ve bu analiz neticesinde model oluşturulur. Daha sonra bu model sıkıştırılan veri ile birlikte depolanır. Bu yaklaşım oldukça basit olmasına rağmen bazı durumlarda modelin depolanması düşük performansa yol açabilir.

Uyarlanır modellerde veri sıkıştırma işlemi yapılırken model dinamik olarak oluşturulup güncellenir. Kodlayıcı ve çözücü başlangıçta önemsiz gibi gözüken bir model oluştururken bilgiler eklendikçe performans açısından güçlenmektedir. Aynı zamanda sıkıştırma işleminde model, veri ile birlikte depolanmaz. Bu da daha karmaşık bir işlem ile birlikte sıkıştırma oranında artışa sebep olmaktadır. Uyarlanır modelli yaklaşım, günümüzde popüler sıkıştırma algoritmalarında sıklıkla kullanılmaktadır.

Yapılan tez çalışması kapsamında uyarlanır yaklaşımlardan olan ve tekrarlı verilerde oldukça iyi sonuçlar elde etmiş Uyarlanır Huffman (Adaptive Huffman) kodlama detaylı bir şekilde ilerleyen başlıklarda anlatılacaktır.

4.2.1.1 *RLE – Run Length Encoding*

Kayıpsız veri sıkıştırmanın en basit yönetimi olarak adlandırılan RLE (Ardışık-tekrarlı Kodlama), veri karakteri ve o karakterin ardışık tekrarlanma sayısının kodlanması ile sıkıştırma işlemi gerçekleştirilir. Özellikle fax cihazlarında geliştirilmiş Huffman kodlama ile birlikte kullanılan RLE, siyah-beyaz renk ayrımına göre kodlama yapmaktadır. Örneğin siyah pikseller S, beyaz pikseller B ile ifade edilecek olursa, siyah-beyaz bir imgeye ait veri, SSSSSBBBSSSBSSSSBBBBB şekilde olabilir. Siyah ve beyaz olan her piksel, tekrar sayısı ile kodlanacak olursa 5S3B3S1B3S5B verisi elde edilir. Yapılan RLE kodlama ile %43'lük bir tasarruf sağlanmış olacaktır.

4.2.1.2 *Sözlük Tabanlı Kodlayıcılar (Dictionary-based Coders)*

Sözlük tabanlı yaklaşımlarda sıkıştırılacak veri ile kodlayıcı tarafından veri içeriği kullanılarak oluşturulan bir grup veri bloğu arasındaki eşleşme ile kayıpsız sıkıştırma gerçekleştirilir. Kodlayıcı, ilk eşleşmeyi bulduğunda veri yapısında bu veri bloğuna ait

indekse karşılık gelen bir yer değiştirme gerçekleştirilir. LZ77 ve LZ78 algoritmaları sözlük tabanlı yaklaşım kullanan algoritmalardır.

LZ77 ve LZ78, Abraham Lempel ve Jacop Ziv tarafından 1977 ve 1978 yıllarında geliştirilen kayıpsız sıkıştırma algoritmalarıdır. LZ1 ve LZ2 olarak da adlandırılan algoritmalar sözlük kodlayıcısı olarak kullanılmaktadır. LZ77 eşleşme işlemini sıralı erişim şeklinde yaparken, LZ78 rasgele erişim ile eşleştirme yapar. Birçok kaynak türünde etkili bir şekilde kullanılabiliyor olmasından dolayı en popüler kayıpsız sıkıştırma algoritması olarak kullanılmaktadır [64].

Terry Welch tarafından LZ78 algoritmasının geliştirilmesiyle oluşturulan LZW sıkıştırma algoritması LZ grubunun en çok kullanılan algoritmasıdır. GIF ve Unix sıkıştırma LZW kullanan sözlük tabanlı sıkıştırma yazılımlarıdır.

4.2.1.3 *Kaynak Tabanlı Sıkıştırma (Context-based Compression)*

Kodlanacak semboller dizisi, sembollerin bağımsız oluşumlarını içermiyorsa, kodlanacak sembolün devamında yer alan sembol bilgisi, kodlanacak sembol hakkında iyi bir fikir verecektir. Bu bilgi dikkate alınacak olursa verilen bir kaynak içerisinde hangi sembolleri daha fazla olasılıkla yer aldığı anlaşılabilir. Bu da olasılık dağılımının eğriliğini arttırmaktadır. Kaynak tarafındaki kodlayıcı, çözücü tarafından biliniyorsa elde edilen olasılık dağılımı sıkıştırma oranını arttırmak için kullanılabilir [62].

Kaynak tabanlı sıkıştırma algoritması kullanan yöntemler şunlardır:

- PPM – Prediction by Partial Matching
- BWT – Burrows-Wheeler Transformation
- ACB – Associative Coder of Buyanovsky
- DMC – Dynamic Markov Compression
- CALIC – Context Adaptive Lossless Image Compression

4.2.1.4 *Entropi Kodlama*

Entropi kodlama, ortamın özel karakteristiklerinden bağımsız olan bir kayıpsız veri sıkıştırma algoritmasıdır. Entropi kodlama orijinal veri üzerinde her sembole bir kod tanımlaması yapar. Sıkıştırma işleminde orijinal veride yer alan semboller, kod karşılıkları ile yer değiştirir ve bu şekilde veri sıkıştırma işlemi gerçekleştirilmiş olur [65]. Sembollere ait kodların uzunlukları, frekans değerlerinin negatif logaritması ile

orantılıdır. Sık kullanılan semboller daha az uzunlukta kodlar ile eşleştirilir.

Entropi kodlama dijital verilerin sıkıştırılması dışında, veri akışları arasındaki benzerliklerin miktarının ölçülmesinde de kullanılabilmektedir.

4.2.1.5 *Huffman Kodlama*

David A. Huffman tarafından 1952 yılında geliştirilen bu yöntem, Basit Entropi Kodlama olarak da adlandırılır. Bilgisayar biliminde Huffman kodlama, kayıpsız veri sıkıştırma yapan bir entropi kodlama algoritmasıdır.

Yapılan tez çalışmasında kullanılacak olan Huffman kodlama ve Uyarlanır yaklaşım, ilerleyen konularda detaylı bir şekilde açıklanmıştır.

4.2.1.6 *Aritmetik Kodlama*

Değişken uzunluklu entropi kodlaması kullanan Aritmetik kodlama, Huffman ile birlikte son zamanlarda sıklıkla kullanılmakta olan kayıpsız sıkıştırma algoritmasıdır. İkili sistemler ve sembol uzunlukları az olan yapılarda sıklıkla kullanılan bir yöntemdir. Entropi değerinin yüksek olduğu durumlarda Huffman kodlamaya göre daha başarılı sonuçlar elde edilebilmektedir. Aritmetik kodlamada her sembol sabit uzunluklu bitlerle temsil edilir. Entropi kodlamanın temelinde olduğu gibi frekans değerleri yüksek olan semboller daha az, düşük frekansa sahip semboller daha fazla bit ile temsil edilirler. Aritmetik kodlama, Huffman kodlamadan farklı olarak bütün mesajı fraction adı verilen bir sayıya kodlar [66].

Huffman kodlama ile karşılaştırıldığında en büyük dezavantajı karmaşık bir yapıya sahip olup daha fazla işlem yükü gerektirmesidir. Birçok alanda Huffman kodlamaya üstünlük sağlayan Aritmetik kodlama, gerçek zamanlı uygulamalarda dezavantajı yüzünden çok fazla tercih edilmemektedir.

4.2.2 Kayıplı Sıkıştırma Algoritmaları

Kayıplı sıkıştırma yöntemlerinde bir takım veri kayıpları göze alınabilmektedir. İnsan gözünün veya diğer duyu organlarının algılamadığı veya güçlükle algılanabilen verileri, orijinal veriden ayrıştırarak yapılan sıkıştırma yönteminin en büyük avantajı, sıkıştırma oranıdır. Kayıplı sıkıştırma, kayıpsız sıkıştırma ile karşılaştırıldığında daha yüksek bir sıkıştırma oranı ile dikkat çekerken, kayıplardan kaynaklı bozulmalar da aynı oranda fark edilebilmektedir.

Dijital birçok uygulamada dikkat çekmeyen kayıplar bir problem teşkil etmemektedir. Dijital bir sesin iletimi veya depolanması aşamasında sesin anlaşılabilir ve rahatsız edici olmaması makuldür. Bu aşamada orijinal ses dosyasında yapılacak kalite optimizasyonu ve sıkıştırma işleminden kaynaklı çeşitli kayıplar göze alınabilir. Sesin kalitesine ve orijinalliğine göre kayıplar farklılık gösterecektir.

Bir ses dosyasında olabileceği gibi dijital video ve resim dizilerinde de kayıplı sıkıştırma yöntemleri sıklıkla kullanılmaktadır.

4.2.2.1 *DCT – Discrete Cosinus Transform, Ayrık Kosinüs Dönüşümü*

Bilginin ifade edildiği ve gösterildiği düzlemden başka bir düzleme çevrilerek o düzlem üzerinde ifade edilmesine *dönüşüm* denir. Dönüşme uğramış bilgi, zaman, frekans ve genlik bilgileri kullanılarak ifade edilir. DCT, kendisini oluşturan sinyallerin kosinüs fonksiyonları şeklinde frekans düzlemine çevrilmesi işlemi olarak tanımlanabilmektedir.

Sinyal üzerindeki değişimler o sinyalin frekans düzlemindeki gösterimi olarak ifade edilir. DCT, sinyalin enerjisini daha küçük bir alana sıkıştırarak, sinyalin daha az veri bitiyle ifade edilmesine olanak sağlamaktadır.

Bilim ve mühendislikte sayısız uygulamada kullanılan DCT algoritması yüksek frekanslı bileşenlerin ihmal edildiği ses ve görüntülerde kayıplı sıkıştırma algoritması olarak kullanılabilmektedir [67].

4.2.2.2 *Fractal Compression, Parçalı Sıkıştırma*

Sıkıştırma işleminde parçalar kullanan algoritma, bir görüntüde yer alan parçaların diğer parçalara benzemesi gerçeğinden yola çıkarak kayıplı sıkıştırma işlemi gerçekleştirilir. Resim parçalarına ayrılır ve her bir parça, parça kodu adı verilen matematiksel veriye dönüştürülür. Dönüşüm işleminin ardından arama ve hesaplama yapılarak sıkıştırma işlemi gerçekleştirilir [68].

4.2.2.3 *Wavelet Encoding, Dalgacık Kodlama*

Dalgacık kodlama, görüntü sıkıştırma için oldukça uygun bir sıkıştırma yöntemi olmasıyla birlikte ses ve video sıkıştırmada da kullanılmaktadır. Dalgacık kodlama kullanan en bilindik uygulama JPEG2000'dir. Bu yöntemde amaç, veriyi mümkün olan en küçük dosya boyutu ile kaydetmektir. Bu sıkıştırma yöntemi kayıplı veya kayıpsız olabilmektedir.

Bu sıkıştırma yönteminde önce dalgacık dönüşüm uygulanır. Bu işlem ile birlikte görüntüde yer alan piksellere ait çok sayıda katsayı üretilir. Daha sonra bu katsayılar sıkıştırılarak işlem adımları tamamlanmış olur.

4.2.2.4 *LPC - Linear Predictive Coding, Doğrusal Tahmin Kodlaması*

Sayısal işaret işleme alanında sesi tanımak için bir kaç algoritma kullanılabilir. Bunların içinde en önemlisi LPC'dir. LPC'nin kullanım kolaylığı ve hafızada az yer kapsaması en belirgin özellikleridir. Doğrusal tahmin kodlaması modelinde temel ilke, ses örneklerinin geçmişteki örneklere bakılarak tahmin edilmesidir. Ses örneğinin, eski örneklerinin doğrusal birleşimi şeklinde olduğu düşünülüp ses sinyalinin karakteristik katsayıları yaklaşık olarak hesaplanır. Elde edilen yaklaşık sonuç ile gerçek değer arasındaki fark, yani hata, minimuma indirilir [69].

4.3 HUFFMAN SIKIŞTIRMA ALGORİTMASI

Huffman algoritması, sıkıştırılacak veri içerisinde sıklıkla tekrarlanan sembolleri daha az bit ile, daha az tekrar eden sembolleri ise daha fazla bit ile kodlayan entropi kodlama tabanlı bir sıkıştırma algoritmasıdır. Genellikle metin dosyaları için kullanılan bu yöntemde semboller, sabit uzunluklu kod yerine sembollerin frekans değerlerine bağlı olacak şekilde değişken uzunlukta bit dizileri ile kodlanır. Sıkıştırma işlemine başlamadan önce kaynak veri taranarak sembollere ait frekans tablosu elde edilir. Bu frekans tablosu her bir sembolün kaynak veri içerisinde kaç adet olduğunu saklamaktadır.

Huffman sıkıştırma algoritması statik ve dinamik olmak üzere ikiye ayrılır. Statik yaklaşım ile yapılan sıkıştırma işlemlerinde frekans tablosuna ihtiyaç duyulurken, dinamik yaklaşımlarda frekans tablosuna ihtiyaç duyulmaz. Bilgisayar sistemlerinde statik yaklaşım sıklıkla kullanılırken, haberleşme ve gerçek zamanlı uygulamalarda dinamik yaklaşım tercih edilmektedir.

Huffman sıkıştırma algoritmasında frekans tablosu oluşturulurken iki farklı yaklaşım dikkat çekmektedir. Bunlardan biri sabit bir alfabe kullanarak frekans tablosu oluşturmaktır. Bir diğer yaklaşımda ise kaynak veri taranarak alfabe ve frekans tablosu dinamik bir şekilde algoritmanın ilk aşamasında oluşturulur. İkinci yaklaşım dilden bağımsız ve daha gerçekçi çözümler üretmesi sebebiyle sıklıkla tercih edilen frekans tablosu yaklaşımıdır. Dilden bağımsız üretilen bu yöntemin dezavantajı, üretilen alfabe ve frekans bilgisinin sıkıştırılan veri kümesi içerisinde saklama zorunluluğudur.

Sıkıştırılan veri ile birlikte frekans tablosunun saklanması küçük çaplı verilerin sıkıştırılmasında genişlemeye sebep olabilecektir.

Frekans tablosu bilinen kaynak veriden, bir sonraki aşamada kod bilgisinin elde edilebilmesi amacıyla *Huffman Ağacı* oluşturulur. Huffman ağacı hangi sembolün hangi kodlar ile temsil edileceğini belirlemeye yarar. Huffman ağacı frekans tablosundan kod bilgisi üreten bir yapıdır. Huffman ağacının üretilmesi ile kaynak veri üzerinde sırayla sembol taraması yapılarak her bir sembole karşılık gelen kod bilgisi kaydedilir. Sıklıkla tekrar eden karakterler Huffman ağacının yapısı gereği daha düşük bitlerle kodlanacağından verimli bir sıkıştırma işlemi gerçekleştirilmiş olur. Algoritmanın son aşamasında frekans tablosu kodlar ile oluşturulan sıkıştırılmış veriye eklenerek sıkıştırma işlemi tamamlanmış olur.

Çözme aşamasında frekans tablosu kodlanmış veriden ayıklanır ve bu frekans tablosu kullanılarak Huffman ağacı oluşturulur. Kod bilgisi ile Huffman ağacı eşleştirilerek her bir koda karşılık gelen sembol kaydedilir ve çözme işlemi gerçekleştirilmiş olur.

4.4 UYARLANIR (ADAPTİVE) HUFFMAN SIKIŞTIRMA ALGORİTMASI

Uyarlanır Huffman kodlama, Dinamik Huffman kodlama olarak da bilinir. Huffman kodlama veride ki olasılıkların bilinmesine dayanmaktadır. Bu bilgi mevcut değil ise, verinin üzerinden ilk geçişte istatistiksel bilgiler toplanır, ikinci geçişte veri kodlanır. Kaynak veriye ait herhangi bir önbilgi olmaksızın, iletilecek sembol kodlarının belirlenmesine, verinin değişen koşullarına göre uyarlanmasına ve bir defada kodlanmasına olanak sağlamaktadır. Bu algoritmayı tek bir geçişli bir işleme dönüştürebilmek için Faller ve Gallagher tarafından uyarlanabilir bir algoritma ortaya konulmuş, daha sonra bu algoritma Knuth ve Vitter tarafından geliştirilmiştir. En büyük faydası gerçek zamanlı olarak kaynak verinin bir defada kodlanmasıdır. Bu yöntemin getirdiği dezavantaj ise iletişim hatalarına karşı oldukça duyarlı olmasıdır. Veri içerisindeki herhangi bir kayıp bütün kodlamayı bozacaktır [70].

Statik Huffman kodlamada, olasılık dağılımı sıkıştırma ve ayrıştırma sürecinde aynı kalmaktadır. Bir kaynak dosyası her defasında tüm kaynak dosyasını okumak gibi aşırı yük gerektiren ön sürçlerden kaçınmak için sadece bir defalığına okunarak kodlanır. Bir alfabe ve olasılık dağılımı önceki deneyimler temel alınarak uygulanır. Böyle bir tahmini modelleme, sıkıştırma kalitesini önemli ölçüde tehlikeye atabilir. Sıkıştırma kalitesindeki

kayıp miktarı, kaynağın olasılık dağılımının tahmini olasılık dağılımından ne kadar farklı olduğuna bağlıdır.

Uyarlanır Huffman Kodlama, kaynak içerik üzerinde istatistiksel tabanlı bir model uygulayarak sıkıştırma oranını arttırmaktadır. Bir alfabe ve frekans tablosu sıkıştırma ve ayrıştırma sürecinde her sembolün tekrar okunmasından sonra yeniden güncellenir. Statik Huffman Kodlama ile karşılaştırıldığında uyarlanır model, kaynağın gerçek durumuna oldukça yakındır.

Yapılan tez çalışması kapsamında geliştilen yazılımın sıkıştırma aşaması Uyarlanır Huffman algoritması kullanılarak gerçekleştirilmiştir. Sıkıştırılmak istenen verinin küçük boyutlu olması ve uygulamanın gerçek zamanlı çalışması uyarlanır yaklaşımın tercih edilmesinde önemli rol oynamıştır.

Yaklaşımın ilk aşamasında kaynak dosyadan bir alfabe elde edilir ve bu alfabede yer alan sembol frekansları her döngünde güncellenir. Aynı zamanda bu sembol ve frekans değerlerine göre Huffman ağacı oluşturulur. Farklı konumlarda bulunan kodlayıcı veya çözücüler, algoritmanın her adımında aynı Huffman ağacını oluştururlar. Bu yaklaşımda Huffman ağacının alıcıya aktarılmasına gerek yoktur.

4.4.1 Sıkıştırma

Sıkıştırma sürecinin ilk adımı, sabit kod (fixed code) tablosunun oluşturulması ile başlar. Sabit kod tablosu, kullanılacak alfabenin ikili olarak kodlanmış şekilde ifade edilmesidir. Tablo, m karakterli bir alfabe için:

$$m = \{2^e + r : \ 2^e > r\} \tag{4.1}$$

formülü ile elde edilir. 26 harfli İngiliz alfabesi için $m = 26$;

$$26 = \{2^e + r : \ 2^e > r\} \tag{4.2}$$

$$26 = \{2^4 + 10 : \ 2^4 > 10\} \tag{4.3}$$

$$e = 4, r = 10 \tag{4.4}$$

Kullanılacak bit sayısı l, sabit kod değeri k olmak üzere:

$$\begin{Bmatrix} 1 \ \leq i \ \leq 2r & \Rightarrow \ l = e + 1, & k = i - 1 \\ 2r < i \leq m & \Rightarrow \ l = e, & k = i - r - 1 \end{Bmatrix} \tag{4.5}$$

Elde edilen veriler doğrultusunda 26 harfli İngiliz alfabesine göre oluşturulmuş sabit kod tablosu Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1 Sabit kod tablosu.

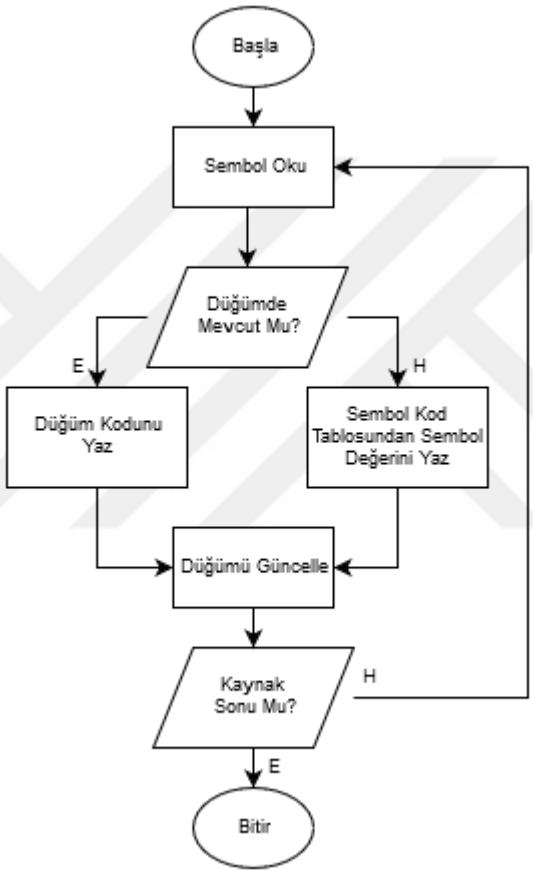
İndis (i)	*Harf*	*Sabit Kod*
1	a	00000
2	b	00001
3	c	00010
4	d	00011
5	e	00100
6	f	00101
7	g	00110
8	h	00111
9	i	01000
10	j	01001
11	k	01010
12	l	01011
13	m	01100
14	n	01101
15	o	01110
16	p	01111
17	q	10000
18	r	10001
19	s	10010
20	t	10011
21	u	1010
22	v	1011
23	w	1100
24	x	1101
25	y	1110
26	z	1111

Sabit kod tablosunun oluşturulmasından sonra Huffman ağacı güncellenerek sıkıştırılmış veri elde edilir. Huffman ağacı, kaynak veri içerisinde bulunan semboller ve bu sembollerin frekans değerine göre oluşturulur. Ağacı oluşturan her bir yapı, düğüm olarak adlandırılır ve ağaç bu düğümlerin birleştirilmesi ile elde edilir.

Şekil 4.2 Başlangıç düğümü

Huffman ağacının ilk düğümü başlangıç düğümü olarak adlandırılır (Şekil 4.2). Kaynak içerisinde bulunan her sembol sırasıyla okunur. Okunan her sembolün düğüm içerisinde yer alma durumu ve frekans değeri baz alınarak Huffman ağacı güncellenir. Eğer sembol Huffman ağacı içerisinde yer almıyorsa NYT adı verilen boş düğüme sembol ve frekans bilgisi işlenir. Huffman ağacı güncellendikten sonra sembolün sabit kod tablosunda yer

alan ikili veri karşılığı sıkıştırılmış veri bloğuna eklenir. Sembolün Huffman ağacı içerisinde yer alması durumunda düğüm kodu, sıkıştırılmış veri bloğuna eklenir ve sembolün bulunduğu düğüm yeni frekans bilgisi ile güncellenir. Sonraki sembol okuma sürecinde eğer sembol Huffman ağacı içerisinde yer almıyorsa sıkıştırılmış veri bloğuna boş düğüm kodu eklenir. Boş düğüm kodunun eklenmesinin ardından sembolün sabit kod tablosunda yer alan ikili kod bilgisi de sıkıştırılmış veri bloğuna eklenerek süreç tamamlanır. Bu işlem kaynak veri uzunluğu kadar devam eder ve sıkıştırılmış ikili veri elde edilmiş olur. Sıkıştırma algoritması akış diyagramı Şekil 4.3'de gösterilmiştir.



Şekil 4.3 Uyarlanır Huffman sıkıştırma akış diyagramı.

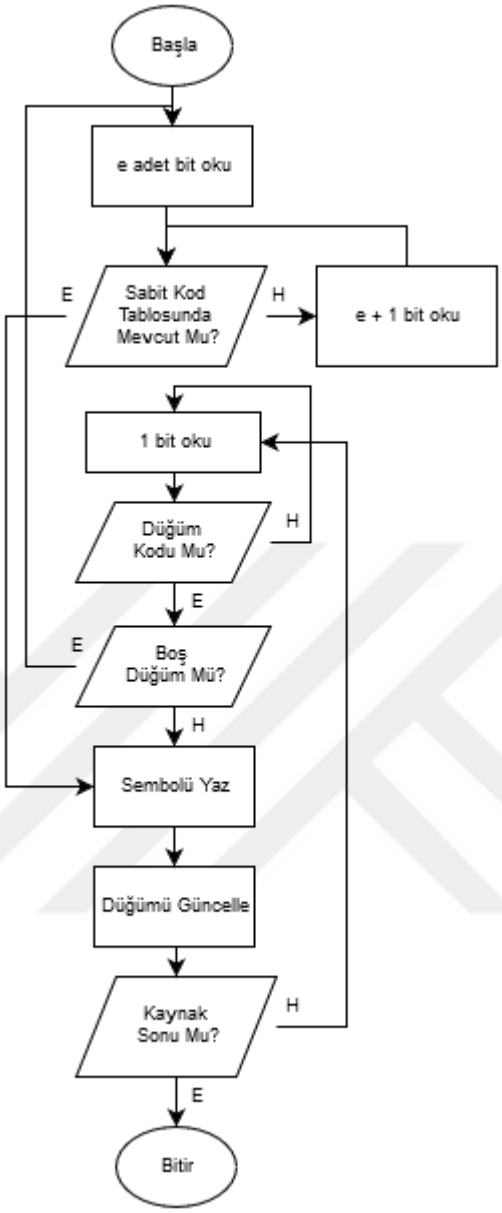
4.4.2 Çözme

Statik Huffman ile kodlanmış veri, içerisinde kaynak veri tarafından oluşturulmuş alfabe ve frekans bilgisi barındırmaktadır. Bu durum kodlanmış verinin boyutunu arttırmaktadır. Uyarlanır Huffman yaklaşımında kodlayıcı ve çözücü ortak bir alfabe kullanır. Ortak alfabe kullanıyor olmasından dolayı aktarım sırasında Huffman ağacı veya alfabe, aktarılacak veriye eklenmez. Ayrıştırma sürecinin ilk adımında ortak alfabeden sabit kod tablosu elde edilir.

Sabit kod tablosunun elde edilmesinden sonra sıkıştırılmış ikili veri üzerinde bit okuma işlemi gerçekleştirilir. İlk okuma işleminde e adet bit okunur. Okunan e adet bit sabit kod tablosunda bir sembole karşılık geliyorsa o sembol ayrıştırılan veri bloğuna yazılır. Eğer okunan e adet bit sabit kod tablosunda bir sembole karşılık gelmiyorsa $e + 1$ adet bit baştan okunur ve sabit kod tablosunda karşılık gelen sembol aranır. Bulunan sembol ayrıştırılmış veri bloğuna eklenir ve Huffman ağacı bu sembol-frekans bilgisi ile güncellenir.

İlk okuma işleminin ardından sıradaki ilk bit okunur. Okunan bit, düğüm içerisinde bir düğüm koduna karşılık geliyorsa, o düğüm koduna ait sembol bilgisi ayrıştırılmış veri bloğuna yazılır. Eğer okunan bit herhangi bir düğüm koduna karşılık gelmiyorsa, bit okuma işlemi karşılık gelene kadar devam eder. Okunan bitler boş düğüm koduna karşılık geliyorsa o bitlerden sonra gelen e veya $e + 1$ bitin sabit kod tablosunda sembol karşılığı aranır ve bulunan sembol ayrıştırılmış veri bloğuna yazılır. Ayrıştırılmış veri bloğuna yazılan sembolün ardından Huffman ağacı sembol-frekans bilgisi ile güncellenir. Bu işlem sıkıştırılmış veri bloğunda bulunan son bit okunana kadar devam eder ve sıkıştırılmış veri elde edilmiş olur.

Ayrıştırma algoritması akış diyagramı Şekil 4.4'de gösterilmiştir.



Şekil 4.4 Uyarlanır Huffman ayrıştırma akış diyagramı

5. ICMP PROTOKOLÜ VE PING

TCP/IP, ABD'de askeri bir proje olarak geliştirilip ilerleyen zamanlarda üniversitelerde kullanılmaya başlayan bir protokol kümesidir. ABD'nin dört bir yerinde dağınık halde bulunan ufak çaplı ağlar, NSFNet adı verilen tek bir omurga altında toplanmış ve sonraları tüm dünyaya yayılmıştır. İnternetin ve ağ sistemlerin gelişimi bu dönemde başlamış ve yaygınlaşmıştır.

Ağları yönetmek, denetlemek ve bağlantı uyumluluğu sağlamak gibi durumlarda protokol adı verilen kurallar kümesi kullanılmaktadır. Bunlardan biri olan TCP/IP, tüm dünya da en yaygın olarak kullanılan protokol kümesidir. Birbirinden farklı ağ yapıları bir arada kullanılmak istendiği zaman devreye TCP/IP protokolü girer. Birbirinden farklı platformlar ve işletim sistemleri yapısında TCP/IP uyumlu yazılımları barındırmaktadır.

TCP/IP protokolü içerisinde yer alan tüm birimler 32 bitlik bir numara ile adreslenirler. Bu numara IP adresi olarak adlandırılmaktadır. 32 bitlik bir IP adresin alabileceği maksimum değer 2^{32} = 4.294.967.296'dır. IP adreslerinin tam sayı şeklinde gösterimi akılda kalıcılık ve kullanım açısından oldukça zordur. Bu yüzden onluk tabanda yazılan IP adresi ilk adım olarak onaltılık sisteme dönüştürülür. Elde edilen onaltılık değer 8'er bitten oluşan 4 bölüme ayrılır. Elimizde bulunan her bir 8 bitlik onaltılık değer tekrardan onluk sisteme dönüştürülerek kullanılabilir IP adresi elde edilir. Bir örnek ile etmek gerekirse:

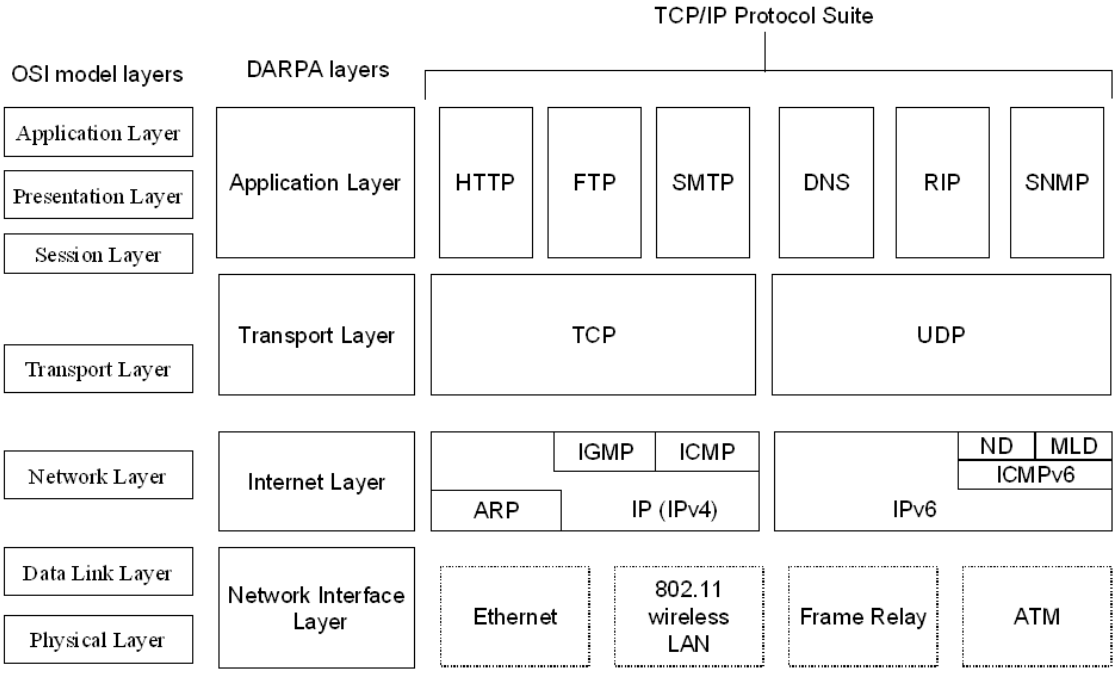
$$\mathrm{IP} = 3.232.235.853 \tag{5.1}$$

$$(3232235853)10 = (\mathrm{C0A8014D})16 \tag{5.2}$$

$$(\mathrm{C0})16 = (192),\ (\mathrm{A8})16 = (168),\ (01)16 = (1),\ (\mathrm{4D})16 = (77) \tag{5.3}$$

$$\mathrm{IP} = 192.168.1.77 \tag{5.4}$$

TCP/IP'de aktarılmak istenen veri bulunduğu katman göz önünde bulundurularak paketlenir ve alıcıda bu paketler ayrıştırılarak orijinal veri elde edilir. TCP/IP protokol kümesi Şekil 5.1'de detaylı bir şekilde gösterilmiştir [71].



Şekil 5.1 TCP/IP Protokol Kümesi.

TCP/IP protokol kümesi ve İnternet Katmanı'nda yer alan IP protokolü, hata bildirme, raporlama veya düzeltme gibi özelliklere sahip değildir. Bağlantısız bir protokol olan IP, UDP üzerinden iki uç arasındaki bağlantı olup olmadığıyla ilgilenmez. ICMP (Internet Control Message Protocol), İnternet Katmanında çalışan ve oluşan sorunları haberleşen birimlere bildiren bir geri besleme mekanizmasıdır. Hataları raporlamak için kullanılan ICMP aynı zamanda uzak sistemler hakkında bilgi toplamak için de kullanılmaktadır. ICMP protokolü IP ile aynı katmanda olmasına rağmen kendisi de IP'i kullanmaktadır. ICMP TCP/IP'nin işlemesine yardımcı olan bir protokol olmasının yanı sıra her hostta mutlaka çalışır [72].

Günümüzde ICMP protokolü;

- TTL (Time to Live) süresi dolduğu zaman paket sahibine geri bildirim yapmak,
- Yok olan paket hakkında paket sahibine geri bildirim yapmak,
- Hata oluşumlarında geri bildirim sağlamak,
- Paket yol seçimi yaparken geri bildirim yapmak amacıyla kullanılmaktadır.

ICMP protokolü içerisinde IP protokolünü barındırmaktadır. Paketin ilk 20 baytlık kısmı IP Protokolüne ait olup kaynak ip adresi, hedef ip adresi, paketin yaşam süresi ve protokol türü gibi önemli bilgiler bu kısımda tanımlanmaktadır. Protokol bilgisinin *1* ayarlanması kullanılan paketin ICMP paketi olduğunu belirtir. *Kaynak* ve *Hedef IP* adres alanıyla birbirleriyle haberleşen birimler tanımlanmış olur. *TTL* ise paketin yaşam süresini belirlemekte ve bu süre zarfında paket yok olur veya hedefe ulaşamazsa kaynak veya hedef birimlere hata geri bildirimi amacıyla kullanılır.

Paketin bir diğer kısmı ICMP protokolü ile alakalı olup 4 temel bölümden oluşmaktadır.

- Type
- Code
- Checksum
- Data

Type alanı mesaj tipini tanımlamaktadır. Mesaj tipi ‘8’ paketin ping paketi olduğunu, ‘0’ ise gelen ping paketine verilen cevabı temsil eder.

.*Type* alanının alacağı değerler Çizelge 5.1 de listelenmiştir.

Çizelge 5.1 ICMP Mesaj Tipleri.

Tip	ICMP Mesajın Tipi
0	Eko yanıt-ping yanıtı (Echo Reply)
3	Hedefe Erişilemedi (Destination Not Reachable)
4	Kaynak Kapatmak (Source Quench)
5	Yeniden Yönlendirme (Redirection Required)
8	Eko yanıt-ping isteği (Echo Request)
9	Yönlendirici tanıtımı
10	Yönlendirici istemi
11	Zaman aşımı—traceroute kullanır (Time to Live Exceeded)
12	Parametre Problemi (Parameter Problem)
13	Timestamp İstemi (Timestamp Request)
14	Timestamp Yanıtı (Timestamp Reply)
15	Bilgi İstemi (Information Request)
16	Bilgi Yanıtı (Information Reply)
17	Addres Maskesi istemi (Address Mask Request)
18	Addres Maskesi yanıtı (Address Mask Reply)

Code alanı hata veya durum bilgisi barındırmaktadır. *Checksum* ise ICMP mesajının 16-bitlik 1'e tümleyenini hesaplar. Son olarak *Data* bölümü mesaj ile ilgili bilgilerin depolandığı paket bölümüdür.

ICMP paketinin genel yapısı Şekil 5.1'de gösterilmektedir. ICMP Payload başlığında bulunan *Data* alanı yapmış olduğumuz çalışmada harici veri taşıyıcısıdır. Sıkıştırılıp şifrelenen veri, *Data* alanına entegre edilerek gizli kanal gerçekleştirilmiş olur.

ICMP packet

	Bit 0 - 7	Bit 8 - 15	Bit 16 - 23	Bit 24 - 31
IP Header (20 bytes)	Version/IHL	Type of service	Length	
	Identification		*flags* and *offset*	
	Time To Live (TTL)	Protocol	Checksum	
	Source IP address			
	Destination IP address			
ICMP Payload (8+ bytes)	Type of message	Code	Checksum	
	Quench			
	Data (*optional*)			

Şekil 5.2 ICMP paket yapısı.

6. ANAHTAR DAĞITIMININ GERÇEKLEŞTİRİLMESİ

Bu bölümde, steganografi, sıkıştırma ve şifreleme teknolojileri kullanılarak gerçekleştirilen uygulama hakkında çalışma ilkelerine dair bilgiler verilip algoritma detaylı bir şekilde anlatılmaktadır.

Bu tez çalışmasında, kablosuz ağlarda anahtar dağıtımı ile ilgili oluşabilecek güvenlik problemleri ve zaman kayıplarını önlemek amacıyla ağ tabanlı uygulama geliştirilmiştir. Uygulama, dağıtılmak üzere kullanılacak anahtar verisini önce sıkıştırıp daha sonra şifreleyerek ağın durumunu test edecek olan ilk ping paketi içerisine gizlenmektedir. Alıcıya aktarılan ping paketi, üzerinde ayrıştırma işlemleri yapılarak anahatar dağıtımı gerçekleştirilmektedir.

Yapılan çalışmada, Microsoft Windows 10 Pro © işletim sistemine sahip bir kişisel bilgisayar (Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, 3101 Mhz, 4 Çekirdek, 16GB RAM) ve bu bilgisayar üzerinde çalışmakta olan bir sanal makine kullanılmıştır. Geliştirilen yazılıma sahip iki bilgisayar (kişisel ve sanal bilgisayar) ping komutu yardımıyla anahtar dağıtımını gerçekleştirmektedir.

Ping paketi ile anahtar dağıtımını gerçekleştirecek olan uygulama, *Visual Studio 2015* © ve *.Net Framework 4.5.2* uygulama geliştirme platformları ile C# programlama dili kullanılarak geliştirilmiştir. Uygulamanın şifreleme aşamasında *System.Security.Cryptography, p*ing paketinin oluşturulması ve gönderiminde *System.Net.Sockets* kütüphanelerinden yararlanılmıştır. Sıkıştırma işlemi için Uyarlanır Huffman algoritması tasarlanmış ve kullanılmıştır.

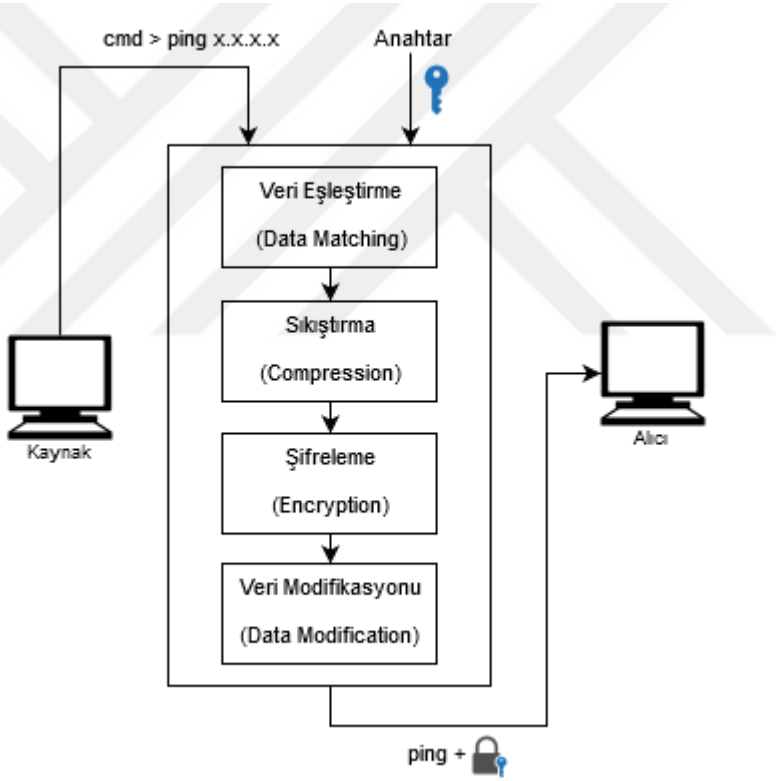
Yazılım, Gönderici ve Alıcı olmak üzere ikiye ayrılmaktadır.

Gönderici tarafında, anahtar verisi kullanılarak oluşturulan ping paketi dağıtılmak üzere belirtilen bir alıcıya gönderilir. Bu aşamada anahtar verisine ait konum haritası, ping paketi içerisinde yer alan varsayılan bilgiler kullanılarak oluşturulur. Oluşturulan bu konum dizisi önce sıkıştırılır (compress) daha sonra ise şifrelenir (encrypt). Son adımda elde edilen veri, paket içerisine yerleştirilerek alıcıya aktarımı gerçekleştirilir.

Alıcı tarafında, ilk olarak paket dinlemesi yapılır. Ping paketi bir ICMP protokolü olduğundan bilgisayara gelen tüm ICMP paketleri dinlenir. İçerisinde anahtar verisini barındıran paket önce bölümlerine ayrıştırılır. Paket içerisinde anahtar veriyi barındıran bölüm, şifre çözme (decrypt) ve ayıklama (decompress) işleminden geçer. Elde edilen konum dizisi gelen ping paketi içerisinde eşleştirilerek anahtar veri elde edilir.

6.1 GÖNDERİM AŞAMASI

Yapılan bu çalışmada kablosuz ağlarda anahtar dağıtım işlemi, algılanamazlık, sağlamlık ve bant genişliği gibi kavramlar dikkate alınarak 4 temel adımdan oluşmaktadır. *Veri Eşleştirme, Sıkıştırma, Şifreleme* ve *Veri Modifikasyonu* gönderim aşamasının temelini oluşturan adımlardır (Şekil 6.1).



Şekil 6.1 Ping paketi hazırlama ve alıcıya gönderilmesi.

6.1.1 Veri Eşleştirme

Bu adımda anahtar veri onaltılık formata dönüştürülüp yine onaltılık formatta olan ping paketi içerisinde bulundukları konum aranmaktadır. Anahtar veriye ait konum bilgileri, bir dizi haline getirilip bir sonraki aşamaya aktarılmaktadır.

Başka bir ifadeyle anahtar veri *K*, ve onaltılık dizi *A* olmak üzere;

$$A = \{a_1, a_2, .. a_n\} = K_{16} \tag{6.1}$$

olur.

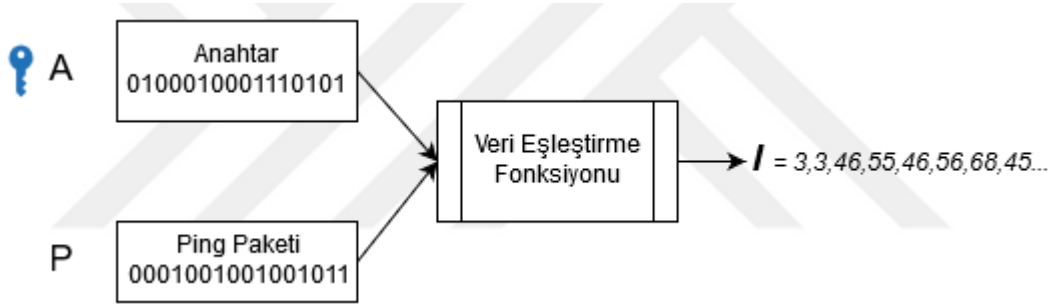
Daha sonra veri modifikasyonuna uğramamış orijinal ping paketini onaltılık bir *P* dizisi şeklinde tanımlanır.

$$P = \{p_1, p_2, .. p_n\} \tag{6.2}$$

Bir sonraki aşamada *A* dizisinin her bir elemanı *P* dizisi içerisinde aranarak konum bilgisi *I* dizisine kaydedilir.

$$I = \{i_1, i_2, .. i_n\} \tag{6.3}$$

Elde edilen *I*, bir sonraki aşamaya sıkıştırılmak üzere aktarılır (Şekil 6.2).



Şekil 6.2 Veri Eşleştirme (Gönderim).

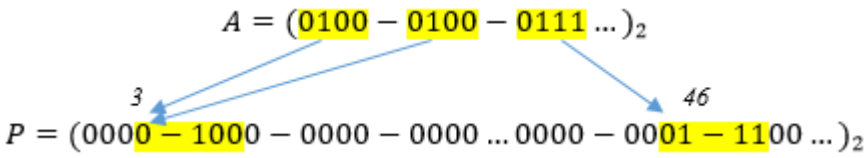
Bir örnek ile ifade etmek gerekirse, dağıtımı yapılacak anahtar veri *Duzce81* olarak tanımlansın. Bu durumda anahtar verinin ikilik tabandaki karşılığı:

$$\begin{aligned} A = (&0100 - 0100 - 0111 - 0101 - 0111 - 1010 - 0110 - 0011 \\ &- 0110 - 0101 - 0011 - 1000 - 0011 - 0001)_2 \end{aligned} \tag{6.4}$$

olur. İçerisinde kaynak ve hedef ip adresi gibi bilgileri barındıran dâhili ping paketinin ikilik tabanda karşılığı şu şekildedir:

$$\begin{aligned} P = (&0000 - 1000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 \\ &- 1000 - 0000 - 0000 - 0001 - 1100 - 0000 - 1010 \\ &- 1000 - 1110 - 0110 - 1000 - 0010 - 0100 - 1111 \\ &- 0111 - 1011 - 1001 - 0000 - 0001 - 0111 - 0000 \\ &- 1000 - 0000 - 0000 - 0000 - 0001 - 0000 - 0000 \\ &- 0000 - 0001 - 0000 - 0000)_2 \end{aligned} \tag{6.5}$$

A ve *P* ikilik dizileri birbirleri ile eşleştirilip konum dizisi elde edilir. *A* dizisinde yer alan her 4 bit *P* dizisinde aranır ve bulunduğu konum *I* dizisine kaydedilir.



Şekil 6.3 Veri Eşleştirme örneği.

Elde edilen konum dizisi şu şekildedir:

$$I = (3,3,46,55,46,56,68,45,68,55,45,4,45,1) \tag{6.6}$$

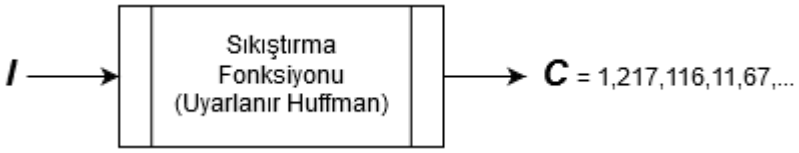
Konum bilgisini barındıran *I*, bir sonraki aşamada sıkıştırılır. Yapılan çalışmalar neticesinde şifreleme işleminden önce sıkıştırma işleminin yapılması sıkıştırma oranını arttırmaktadır. Şifrelenmiş veri üzerinde birbirini tekrar eden sembollerin azlığı sıkıştırma oranını düşürmektedir. Bu yüzden konum bilgisi barındıran dizinin sıkıştırılması tercih edilmiştir.

6.1.2 Sıkıştırma

Bu aşamada anahtar veri kullanılarak oluşturulmuş *I* konum dizisi, gizli kanal aracılığıyla aktarılmak istenen anahtar veri uzunluğundan daha fazladır. Bu sebepten dolayı sıkıştırma, gönderim aşamasının önemli bir adımını oluşturmaktadır. Sıkıştırma algoritmaları genellikle birbirini tekrar eden veriler üzerinde oldukça başarılı sonuçlar elde eder. Bu yüzden daha kısa uzunluk ve tekrar sayısı az olan anahtar verisi yerine, anahtar verinin paket içerisindeki konumu belirten dizi üzerinde sıkıştırma işlemi gerçekleştirilmiştir. Sıkıştırma işlemi, geniş kullanıma sahip ve güncel bir sıkıştırma algoritması olan Uyarlanır Huffman (Dinamik Huffman) ile gerçekleştirilmiştir. Bir önceki adımda elde edilen *I*, sıkıştırılarak aktarılacak veri boyutu minimuma düşürülmektedir. Bir başka ifadeyle;

$$C = Compress(I) \tag{6.7}$$

Sıkıştırma işleminin ardından elde edilen *C*, aktarımın güvenli olabilmesi için bir sonraki aşamada şifreleme işlemine alınır (Şekil 6.4).



Şekil 6.4 Sıkıştırma.

Örnekte ele alınan *Duzce81* anahtar verisi için elde edilen *I* konum dizisini sıkıştırma işlemine alınır. Sıkıştırma işleminin ardından elde edilen *C* şu şekildedir:

$$C = (1,217,116,111,67,136,137,139,119,113,2,60,1,64) \tag{6.8}$$

Yapılan sıkıştırma işlemine ait süre, *System.Diagnostics* kütüphanesinde yer alan *Stopwatch* fonksiyonu ile belirlenmiştir.

Çizelge 6.1 Farklı anahtarlara ait sıkıştırma verileri.

Anahtar	I (byte)	C (byte)	Oran (%)	Süre (ms)	Tik
Duzce81	14	14	1	11	35049
MerhabaDunya	24	20	16,67	11	34788
cagatayay	18	13	27,78	11	33268

Çeşitli anahtar değerlerine ait sıkıştırma verileri Çizelge 6.1'de gösterilmiştir. Farklı uzunluklarda ve içeriğe sahip anahtar verilerinde sıkıştırma oranı değişirken işlem süresi sabit oranda kalmaktadır. Sıkıştırma süresi yaklaşık 11ms olarak ölçülmüştür. Tik süresi *Stopwatch* fonksiyonu tarafından üretilen en küçük zaman birimidir.

6.1.3 Şifreleme

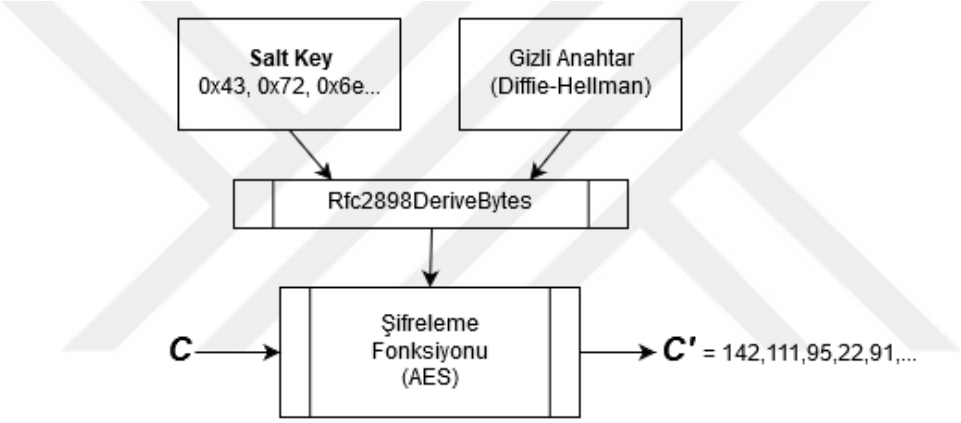
Sıkıştırılmış konum dizisi C, aktarılmadan önce güvenli haberleşme için şifrelenir. Bu çalışmada günümüzde yaygın olarak kullanılan şifreleme yöntemlerinden biri olan AES algoritması kullanılarak ağ üzerinden güvenli parola aktarımı sağlanması amaçlanmıştır. AES blok şifreleme algoritmasını kullanılır. Veriler 4×4'lük diziler (matris) hâlinde bloklandıktan sonra uzunluğu en az 128 bit olan anahtarlar kullanılarak şifreleme gerçekleştirilir. Yüksek hız ve düşük hafıza kullanımı nedeniyle yapılan bu çalışmada AES tercih edilmiştir [73].

Bir başka ifadeyle açıklamak gerekirse;

$$C' = Encypt(C) \tag{6.9}$$

AES, simetrik bir şifreleme algoritması olduğu için gizli anahtara ihtiyaç duyar. Alıcının şifreli veriyi çözebilmesi için bu anahtara sahip olması gerekir. Açık haberleşme kanalları kullanıldığından anahtar paylaşımının da güvenli bir şekilde yapılması gerekmektedir. Diffie-Hellman algoritması, güvensiz medyalar arasında gizli anahtar üreten özel bir yöntemdir.

Yapılan tez çalışması kapsamında Diffie-Hellman algoritması kullanılarak *Qwe123** gizli anahtarı üretilmiş ve bu anahtar kullanılarak şifreleme işlemi gerçekleştirilmiştir (Şekil 6.5).



Şekil 6.5 Şifreleme.

Yapılan örnek çalışmada, sıkıştırılan *C* dizisi ile elde edilen C' şu şekildedir:

$$C' = (142,111,95,22,91,110,15,107,7,185,247,24,119,172,155,42) \tag{6.10}$$

AES ile şifrelenen C', ping yazılımı ile aktarıma hazırdır. Gönderimin son aşamasında şifrelenmiş veri, paket içerisine gizlenip aktarım işlemi gerçekleştirilir.

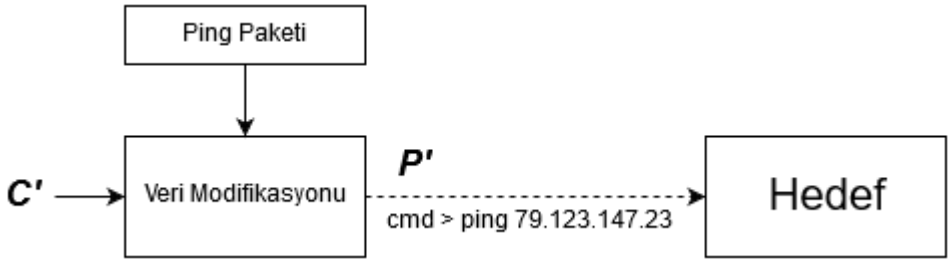
6.1.4 Veri Modifikasyonu

Bu aşamada elimizde bulunan C' verisi ping paketi içerisinde *data* alanına yerleştirilir. Bir başka ifadeyle açıklamak gerekirse:

$$P' = Concatenate(P, C') \tag{6.11}$$

Sıkıştırılmış ve şifrelenmiş konum dizisi C', hazırlanan ping paketi ile veri modifikasyonu aşamasında birleştirilir. Birleşim işleminin ardından paket alıcıya aktarılır ve anahtar

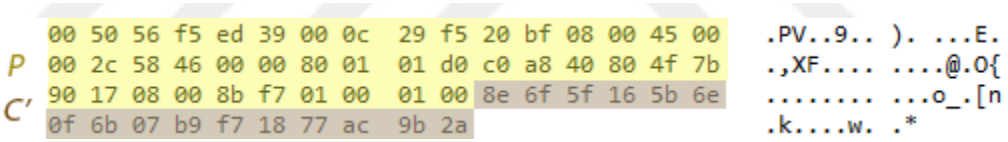
teslimi gerçekleştirilmiş olur. Alıcı tarafında çeşitli aşamalardan geçecek olan ping paketi, çözümlenerek anahtar veri elde edilmiş olacaktır (Şekil 6.6).



Şekil 6.6 Veri Modifikasyonu.

Data alanı ping paketi içerisinde rasgele büyüklükte veri göndermek için kullanılan bir alandır. TCP ile gönderilebilecek maksimum paket boyutu 64K (65535 bayt) şeklindedir. Yapılan çalışmada, elde edilen harici veri uzunlukları 16-32 bayt aralığındadır. Ping paketine harici verinin eklenmesiyle veri aktarım işlemi gerçekleştirilmiş olur. Anahtar verinin konumlandırma yöntemiyle ping paketine entegrasyonu ağ steganografisi'nin temel ilkelerinden algılanamazlığı ön plana çıkarırken, şifreleme ile sağlamlık, son olarak sıkıştırma ile etkin bant genişliği kullanımı ele alınmıştır.

Veri modifikasyonu sonrası oluşan ping paketimiz Şekil 6.7'de gösterilmiştir.



Şekil 6.7 Veri modifikasyonu sonrası oluşan ping paketi.

Stego-nesne olan ping paketi 44 bayttan oluşmaktadır. İlk 20 baytlık kısım ip başlığına ait olup paket yaşam süresi (TTL), kaynak ip adresi, hedef ip adresi ve kullanılacak protokol bilgisini içerisinde barındırmaktadır. Sonraki 24 bayt ICMP protokolüne ait olup ilgili protokole ait bilgiler içermektedir. 24 baytlık veri içerisindeki son 16 bayt, *data* alanı ait olup ilk 28 baytlık veri kullanılarak sıkıştırılmış ve şifrelenmiş anahtar verimizin konum bilgisini barındırmaktadır.

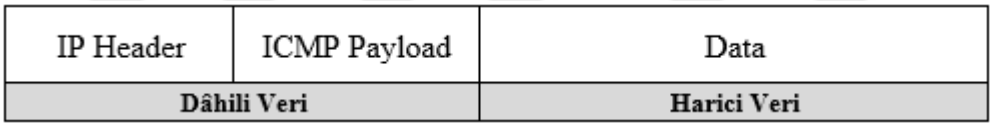
Oluşturulan ping paketi ile ilgili detaylı bilgi Çizelge 6.2'de belirtilmiştir.

Çizelge 6.2 Stego-nesne.

<table>
<tr><th>Version</th><th>Header Length</th><th>DSCP</th><th>Total Length</th><th>Identification</th><th>Flags</th><th>Fragment Offset</th><th>TTL</th></tr>
<tr><td rowspan="2">4</td><td>5</td><td rowspan="2">00</td><td>00 2c</td><td rowspan="2">58 46</td><td rowspan="2">00</td><td rowspan="2">00 00</td><td>80</td></tr>
<tr><td>20 bayt</td><td>44</td><td>128</td></tr>
<tr><th>Protocol</th><th>Header Checksum</th><th>Source</th><th>Destinition</th><th>Type</th><th>Code</th><th>Checksum</th><th>Quench</th></tr>
<tr><td>01</td><td rowspan="2">01 d0</td><td>4f 7b 90 17</td><td>c0 a8 40 80</td><td>08</td><td rowspan="2">00</td><td rowspan="2">8b f7</td><td rowspan="2">01 00
01 00</td></tr>
<tr><td>ICMP</td><td>79.123.144.23</td><td>192.168.64.128</td><td>Echo = Ping</td></tr>
<tr><th colspan="8">DATA</th></tr>
<tr><td colspan="8">8e 6f 5f 16 5b 6e 0f 6b 07 b9 f7 18 77 ac 9b 2a</td></tr>
</table>

6.2 ÇIKARIM AŞAMASI

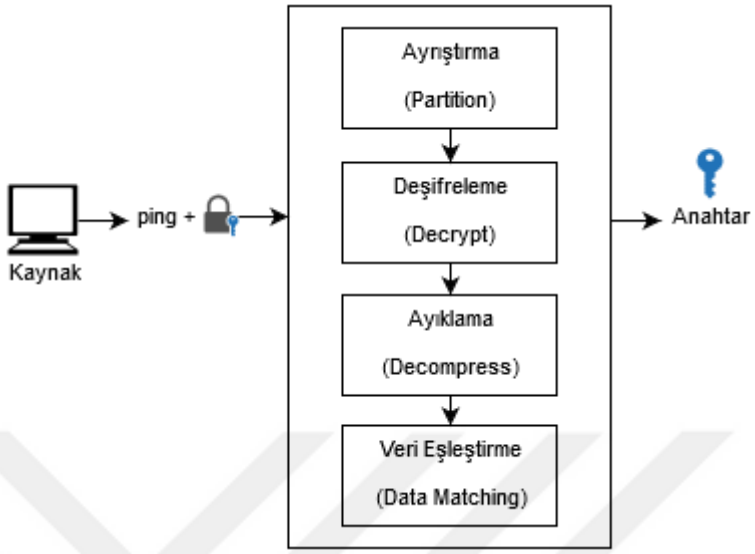
Gönderici tarafından ilgili alıcıya ping paketi aracılığıyla aktarılan harici veri, alıcı tarafında uygulanan birkaç işlem sonucunda tekrar elde edilir. Alıcıya aktarılan ping paketi içerisinde harici veri ile birlikte birçok veri bloğu bulundurmaktadır (Şekil 6.8).



Şekil 6.8 Ping paketi veri blokları.

Paket bloğu içerisinde, kaynak ve hedef bilgisi, yaşam süresi, protokol türü gibi birçok veri bulundurmaktadır. Anahtar verinin elde edilmesi işleminde ilk adım harici verinin paket içerisinden ayıklanmasıdır. Ayıklanan harici veri gönderim aşamasında sıkıştırılıp şifrelendiğinden kullanılan algoritmaların çözme ve ayrıştırma adımları uygulanarak bir konum dizisi elde edilir. Elde edilen bu dizi, onaltılık formata dönüştürülmüş dâhili veri içerisinde bulunan gizlenmiş anahtarın konum bilgisini barındırmaktadır. Konum bilgisi bilinen anahtar veri, dâhili paket içerisinde aranır ve arama işlemi sonucunda anahtar elde edilmiş olur.

Çıkarım aşaması 4 temel adımdan oluşmaktadır. İlgili alıcıya aktarılan paket veri, sırasıyla *Ayrıştırma, Şifre Çözme, Ayıklama, Veri Eşleştirme* adımlarından geçerek anahtar veri elde edilir (Şekil 6.9).



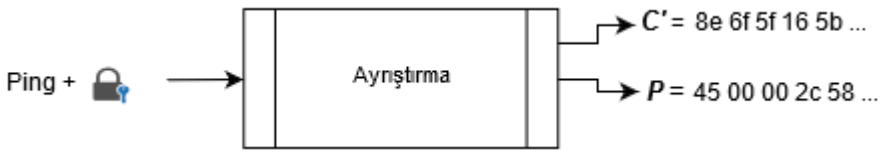
Şekil 6.9 Çıkarım aşaması ve anahtar verinin elde edilmesi.

6.2.1 Ayrıştırma

Aktarılan paket veri içerisinde dâhili birçok veri bulundurmaktadır. Harici verimizle birlikte kaynak ve hedef ip adresi, protokol türü, paket yaşam süresi, servis türü, servis tipi, sürüm bilgisi, mesaj tipi gibi veri blokları ICMP paketini oluşturmaktadır. Aktarılan paket içerisindeki harici veri, sıkıştırılmış ve şifrelenmiş anahtar verinin konum bilgisini içerisinde barındırırken dâhili veri, konum bilgisi bilinen anahtar veriyi içerisinde barındırmaktadır. Çıkarımın ilk aşamasında, dâhili ve harici veri ayrıştırılmaktadır. Ayrıştırma işlemi sonucunda sıkıştırılmış veri elde edilerek ilk aşama tamamlanmış olur. Bir başka ifadeyle şifrelenmiş veri C':

$$C' = Seperate(P, P') \tag{6.12}$$

olur. Elde edilen C', bir sonraki aşamada *Şifre Çözme* işlemine alınır (Şekil 6.10).



Şekil 6.10 Ayrıştırma.

Gönderici aşamasında hazırlanan ping paketini dahili ve harici veri bloklarına ayrıştırılacak olursa:

$$P = (08\ 00\ 45\ 00\ \ 00\ 2c\ 58\ 4b\ 00\ 00\ 80\ 01\ 01\ cb\ c0\ a8\ 40\ 80\ 4f\ 7b\ 90\ 17\ 08\ 00\ 8b\ f7\ 01\ 00\ 01\ 00\ 8e\ 6f\ 5f\ 16\ 5b\ 6e\ 0f\ 6b\ 07\ b9\ f7\ 18\ 77\ ac\ 9b\ 2a\ ) \quad (6.13)$$

$$P' = (08\ 00\ 45\ 00\ \ 00\ 2c\ 58\ 4b\ 00\ 00\ 80\ 01\ 01\ cb\ c0\ a8\ 40\ 80\ 4f\ 7b\ 90\ 17\ 08\ 00\ 8b\ f7\ 01\ 00\ 01\ 00\ ) \quad (6.14)$$

$$C' = (8e\ 6f\ 5f\ 16\ 5b\ 6e\ 0f\ 6b\ 07\ b9\ f7\ 18\ 77\ ac\ 9b\ 2a) \quad (6.15)$$

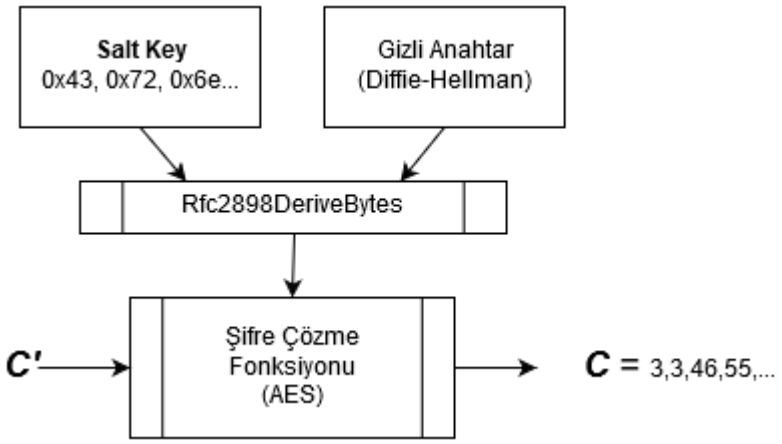
Yapılan işlem sonucunda dâhili veri P' ve harici veri C' olarak ayrıştırılır.

6.2.2 Şifre Çözme

Ayrıştırılıp şifrelenmiş C' verisi ikinci aşamada çözülerek anahtar veriye ait sıkıştırılmış konum dizisi elde edilir. Gömme aşamasında AES ile şifrelenmiş veriye, aynı algoritmanın ters adımları uygulanarak çözme işlemi gerçekleştirilir (Şekil 6.11). Bir başka ifadeyle sıkıştırılmış konum dizisi C:

$$C = Decrypt(C') \quad (6.16)$$

olur. Elde edilen *C* dizisi, bir sonraki aşamada *Ayıklama* işlemine alınır ve anahtar veriye ait konum dizisi elde edilir.



Şekil 6.11 Şifre Çözme.

Şifre çözme aşaması, şifreleme aşamasında olduğu gibi bir gizli anahtara ihtiyaç duymaktadır. Şifrelemede olduğu gibi Diffie-Hellman algoritması kullanılarak gizli anahtar *Qwe123** üretilir. Gizli anahtar ve şifrelenmiş harici veri kullanılarak şifre çözme işlemi gerçekleştirilir.

Örnek çalışmaya ait çözülmüş şifre C dizisi şu şekildedir:

$$C = (1,217,116,111,67,136,137,139,119,113,2,60,1,64) \quad (6.17)$$

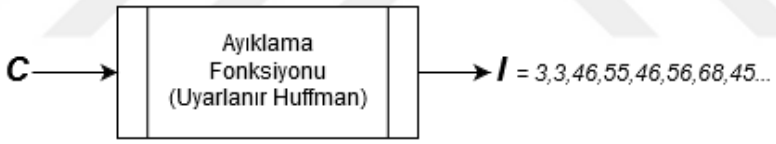
6.2.3 Ayıklama

Çözülen harici veri, çıkarımın üçüncü aşamasında ayıklanır. Ayıklama işlemi, sıkıştırılmış veriden anahtar veriye ait konum bilgisini barındıran dizinin elde edilmesi olarak tanımlanabilir. Uyarlanır Huffman yöntemi ile sıkıştırılan veriden aynı algoritmanın ayıklama adımları uygulanarak konum dizisi elde edilir (Şekil 6.12). Birbirine bağlı düğümler ile bir veri ağacı oluşturulan Uyarlanır Huffman yönteminde yine aynı ağaç kullanılarak ayıklama işlemi yapılır. Bir başka ifade ile ayıklanmış konum dizi I;

$$I = Decompress(C) \quad (6.18)$$

olur.

İşlem basamakların son aşamasında I konum dizisine *Veri Eşleştirme* işlemi uygulanarak anahtar veri elde edilir.



Şekil 6.12 Ayıklama.

Örnek çalışmada, *Ayıklama* işlemine tabi tutulan C verisinden elde edilen I konum dizisi şu şekildedir:

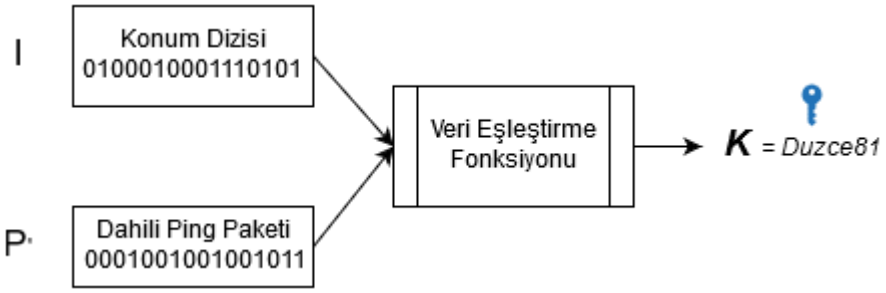
$$I = (3,3,46,55,46,56,68,45,68,55,45,4,45,1) \quad (6.19)$$

Harici veriden şifre çözme ve ayıklama işlemleri sonrası elde edilen I, ping paketine ait dâhili veri içerisine gizlenmiş anahtarın konum bilgisini içermektedir. Bir sonraki aşamada yapılacak *Veri Eşleştirme* yöntemiyle anahtar veri elde edilmiş olacaktır.

6.2.4 Veri Eşleştirme

Ayrıştırma aşamasında elde edilen dâhili veri, şifre çözme ve ayıklama işlemi sonucunda elde edilen konum dizisi ile eşleştirilir. Elimizde bulunan dâhili veri, eşleştirme işlemi için onaltılık formata dönüştürülür.

Dönüştürülen veri bloğu konum dizisi içerisinde yer alan sayısal veri (indeks) ile eşleştirilerek anahtar veri elde edilir (Şekil 6.13).



Şekil 6.13 Veri Eşleştirme (Çıkarım).

Bir başka ifadeyle onaltılık dâhili veri P', konum dizi I olmak üzere;

$$P_{16} = \{p_1, p_2, p_3 \dots p_n\} \tag{6.20}$$

$$I = \{i_1, i_2, i_3 \dots i_n\} \tag{6.21}$$

olur. I dizisi içerisinde yer alan her bir konum değerine karşılık gelen P'_{I_i} değeri bulunarak bir onaltılık A verisi elde edilir. Mevcut verinin metin türüne dönüştürülmesiyle anahtar verinin ping paketi ile dağıtım işlemi tamamlanmış olur.

$$K = A_{16} = \{a_1, a_2, a_3 \dots a_n\} \tag{6.22}$$

Yapılan örnek çalışmada ping paketine ait P' ikilik tabanda yazılacak olursa:

$$\begin{aligned} P' = (&0000 - 1000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 \\ &- 1000 - 0000 - 0000 - 0001 - 1100 - 0000 - 1010 \\ &- 1000 - 1110 - 0110 - 1000 - 0010 - 0100 - 1111 \\ &- 0111 - 1011 - 1001 - 0000 - 0001 - 0111 - 0000 \\ &- 1000 - 0000 - 0000 - 0000 - 0001 - 0000 - 0000 \\ &- 0000 - 0001 - 0000 - 0000)_2 \end{aligned} \tag{6.23}$$

Konum dizisine ait eşleştirme işleminden sonra elde edilen ikilik veri ASCII dönüşüm ile anahtar veriye dönüştürülür ve bu şekilde gönderim aşamasında ping paketi içerisine gizlenen anahtar veri *Duzce81* elde edilmiş olur.

Yapılan örnek çalışmanın Veri Eşleştirme aşamasına ait sonuçlar Çizelge 6.3'de listelenmektedir.

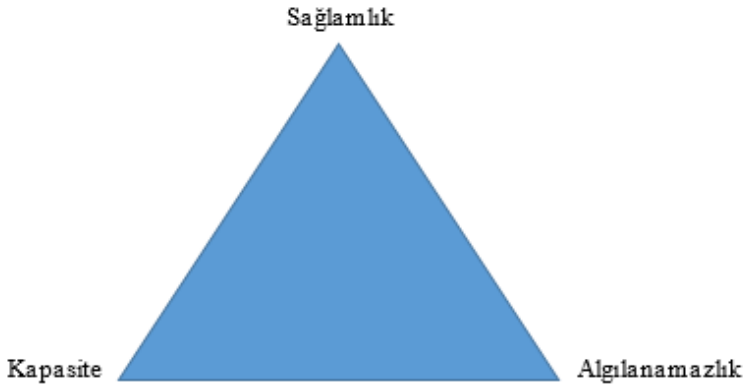
Çizelge 6.3 Veri Eşleştirme ve anahtar verinin elde edilmesi.

	I	***P′***	**Hex**	**ASCII**
1	3	0100	4	D
2	3	0100	4	
3	46	0111	7	u
4	55	0101	5	
5	46	0111	7	z
6	56	1010	a	
7	68	0110	6	c
8	45	0011	3	
9	68	0110	6	e
10	55	0101	5	
11	45	0011	3	8
12	4	1000	8	
13	45	0011	3	1
14	1	0001	1	

Bu bölümde ping paketi ile anahtar dağıtımı aşamaları detaylı bir şekilde anlatılmış ve örnek bir anahtar dağıtım işlemi gerçekleştirilmiştir. Yapılan tez çalışmasına ait deneysel sonuçlar bir sonraki bölümde anlatılacaktır.

7. DENEYSEL ÇALIŞMALAR

Bir steganografik sistem farklı bakış açılarıyla değerlendirilmektedir. Bunlar bilgi gizlenen örtü verisinin (cover object) ne kadar değiştiği, bilgi saklama kapasitesi ve sistemin dayanıklılığının ne kadar olduğudur [25]. Bir steganografik sistemin başarımını değerlendirmek için bu üç kritere bakılması gerekmektedir. Bu kriterleri steganografi ve steganaliz alanında çalışmaları olan bilim insanı Jessica Fridrich ortaya atmıştır. Sağlamlık, kapasite ve algılanamazlık kriterleri arasındaki ilişki Şekil 7.1'de gösterilmiştir.



Şekil 7.1 Fridrich üçgeni.

Günümüzde birçok steganografik metot vardır ve bu metotların başarımının ölçülmesi için Fridrich üçgeni referans alınmalıdır. Her steganografik yöntem algoritmik olarak farklı metotlar izlediği için farklı analiz metotları geliştirilmiştir. Bundan dolayı her metodun kendine özgü bir steganaliz metodu bulunmaktadır. Fridrich üçgenine göre geliştirilmekte olan metot, herhangi bir kriter üzerinde daha başarılı olması durumunda diğer kriterlerden ödün vermek durumundadır. Örneğin daha fazla bilgi gizleme yapılmak istenen bir yöntemde kullanılan kapasite artarken, aynı oranda algılanamazlık ve sağlamlık düşecektir.

Şifrelenmiş bir sistemin güvenliği entropi kavramı ile ölçülmektedir. Yüksek bit miktarı ve entropi değerine sahip verilerin kırılması oldukça zordur. Aynı zamanda entropi, bir sistemin ne kadar düzensiz olduğunu ifade etmekte kullanılır. Bir örnek ile açıklayacak olursak masada duran bir bardağın entropi değeri sıfır iken, bu bardağın yere düşüp parçalanmasıyla entropi değeri tavan yapacaktır.

Entropi kavramı ilk olarak Shannon tarafından bilgisayar bilimlerinde ve iletişiminde kullanılmıştır. Shannon Entropisi olarak da adlandırılan bu kavrama göre bir mesajı kodlamak için gereken en kısa ihtimallerin ortalama değeri, alfabede bulunan sembollerin logaritmasının entropiye bölümüdür. Alfabemizde 256 karakter olduğunu varsayarsak bu sayının logaritmasını mesajın entropisine böleriz. Mesajdaki değişim ne kadar fazla ise o kadar fazla koda ihtiyacımız olacaktır. Bilgisayar bilimleri açısından bir tanım yapmak gerekirse elimizdeki veriyi kaç bit ile kodlayabileceğimize entropi adı verilir. Bir bilginin entropisi hesaplanırken;

$$H = -\sum_{i=0}^{N-1} p_i \log_2 p_i \tag{7.1}$$

formülünden yararlanılmaktadır. Formüle göre, birbirinden farklı sembollerin sayısı N, o sembollere ait ihtimaller ise p olarak ifade edilmektedir.

Geliştirilen uygulamada anahtar veriye ait her bir konum bilgisi, dizi içerisinde 4 bit uzunlukta saklanmaktadır. Konum bilgisi üzerinde şifreleme ve sıkıştırma işlemleri yapıldığından entropi değeri maksimum 4 olacaktır. Farklı uzunluklardaki anahtar verilerine ait sıkıştırma, şifreleme ve entropi değerleri Çizelge 7.1'de listelenmiştir.

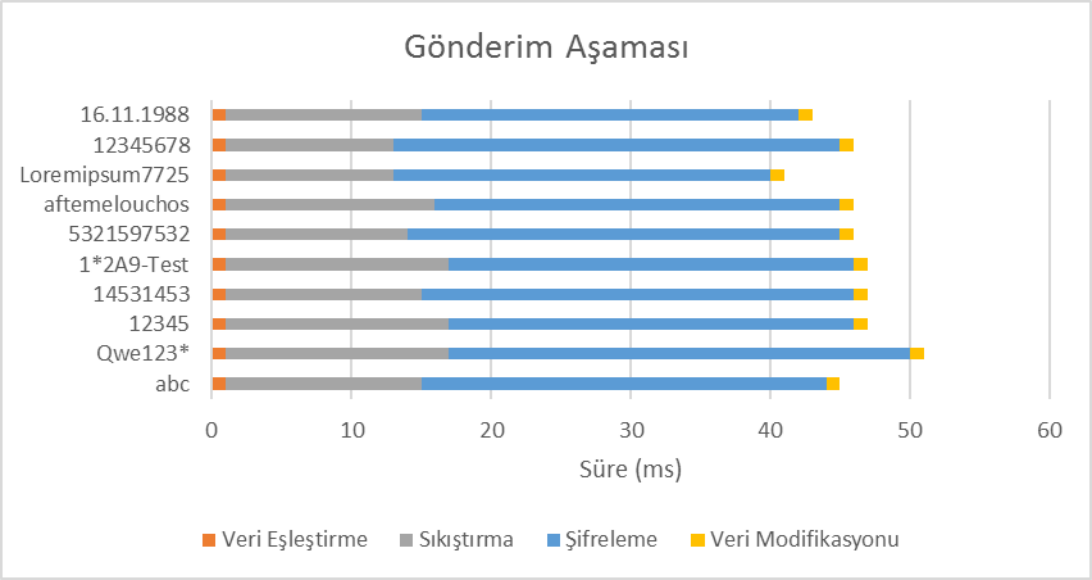
Çizelge 7.1 Farklı anahtar uzunluklarına ait süre, kapasite ve entropi değerleri.

	Anahtar	**t_I** *(ms/tick)*	**I** *(byte)*	**t_C** *(ms/tick)*	**C** *(byte)*	**$t_{C'}$** *(ms/tick)*	**C'** *(byte)*	**P** *(byte)*	**Entropi**
1	Abc	0/292	6	14/44319	6	29/89301	16	58	3,65
2	Qwe123*	0/791	14	16/48738	13	33/100681	16	58	3,62
3	12345	0/345	10	16/50517	8	29/89365	16	58	3,73
4	14531453	0/503	16	14/43112	9	31/96844	16	58	3,69
5	1*2A9-Test	0/565	20	16/49068	19	29/89238	32	74	3,77
6	5321597532	0/709	20	13/40998	12	31/93908	16	58	3,78
7	aftemelouchos	0/935	26	15/46351	20	29/89940	32	74	3,79
8	Loremipsum7725	0/924	28	12/38920	23	27/64310	32	74	3,91
9	12345678	0/660	16	12/37797	13	32/969993	16	58	3,50
10	16-11-1988	0/838	20	14/44467	14	27/82981	16	58	3,65

Elde edilen deneysel sonuçlara göre, veri eşleştirme ve veri modifikasyonu aşamaları yaklaşık 1ms'de gerçekleşirken, karmaşık yapıya sahip sıkıştırma ve şifreleme aşamaları anahtar uzunluklarına bağlı olarak ortalama 28ms'de gerçekleşmektedir.

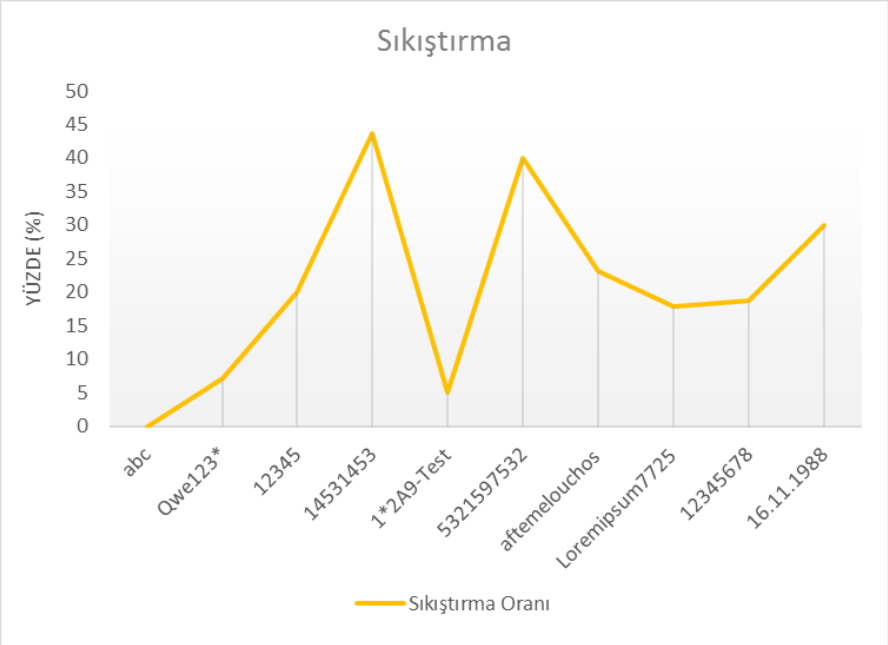
Gönderim aşamasına ait grafiksel gösterim Şekil 7.2'de gösterilmiştir. Anahtar dağıtımına hazırlanan ping paketine ait toplam işlem süresi ortalama olarak 45ms ölçülmüştür. Kullanılan donanımsal sistemlerin performansına bağlı olarak bu işlem süreleri farklılık gösterebilir. Günümüze ait donanımsal performans ve gerçek zamanlı

uygulama verileri dikkate alındığında kaydedilen sürelerin yeterli olduğu gözlenmiştir.



Şekil 7.2 Gönderim aşamasına ait işlem süreleri (ms).

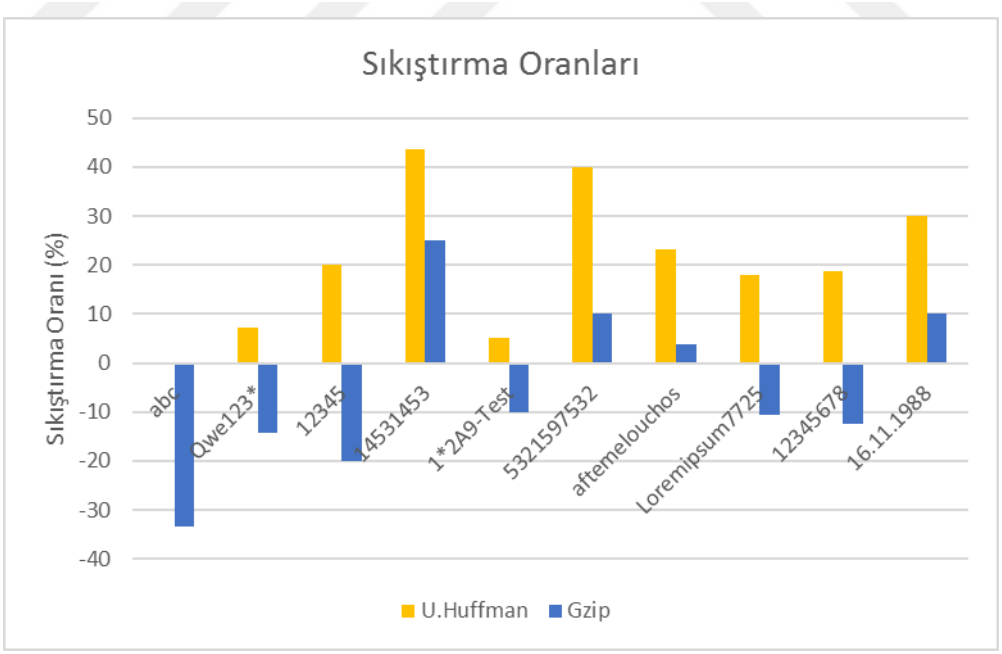
Çizelge 7.1'e göre ölçülen kapasite değerleri incelendiğinde, birbirini tekrar eden semboller içeren anahtar verilene ait sıkıştırma oranının yüksek, karmaşık yapıya sahip ve aynı zamanda anahtar uzunluğu düşük olan verilerde sıkıştırma oranının düşük olduğu gözlenmiştir. Sıkıştırma oranının düşük olması, sıkıştırma işleminin başarısız olduğu anlamına gelmemelidir. Şekil 7.3'de belirtilen sıkıştırma oranları incelendiğinde karmaşık yapıda olan tekrarsız *1*2A9-Test* ve uzunluğu 3 bayt olan *abc* anahtar verilerinde sıkıştırma oranı düşük kalıyorken, *14531453* ve *5321597532* gibi tekrarlı semboller içeren anahtar verilerinde bu oran oldukça yüksektir.



Şekil 7.3 Sıkıştırma oranları (%).

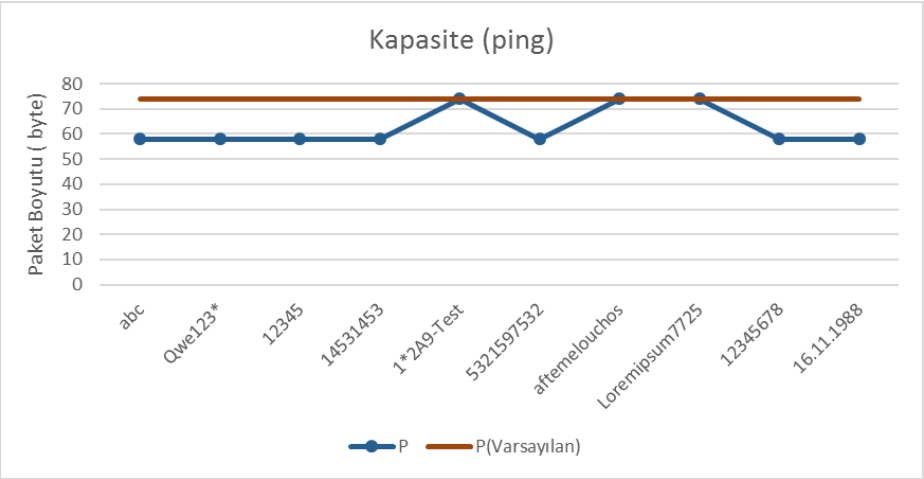
Uygulamanın gönderim aşaması geliştirilirken konum bilgisi elde edildikten sonra şifreleme ve sıkıştırma işlemleri gerçekleştirilmiştir. Sıkıştırma işleminin şifrelemeden sonra yapılması kapasite değerini olumsuz etkilemiştir. Şifrelenmiş veride tekrarsız sembollerin olması sıkıştırma oranını düşürmektedir. Aynı zamanda şifreleme sonrası elde edilen sabit uzunluktaki paket boyutu, sistem dışı kullanıcıların paketi ele geçirmeleri halinde şifreyi tahmin edebilmelerine olanak sağlamayacaktır. Sıkıştırma işlemi en son yapıldığı zaman paket boyutu değişken olabileceğinden, şifrenin kaç sembolden oluştuğu veya tekrarlı sembol içerip içermediği tahmin edilebilir. Bu da sağlamlık açısından çalışmanın kırılgan olabileceği anlamına gelir.

Yapılan çalışmanın sıkıştırma aşamasında farklı algoritmalar test edilerek performans analizi yapılmıştır. Popüler bir veri sıkıştırması olan Gzip sıkıştırma ile karşılaştırıldığında Uyarlanır Huffman, sıkıştırma oranı dikkate alındığında yüksek, süre dikkate alındığında düşük performans göstermiştir. Şekil 7.4'de yer alan grafik incelendiğinde, bazı örneklerin negatif yönde sıkıştırma oranına sahip olduğu gözlenmiştir. Negatif yönde sıkıştırma oranı, sıkıştırılmış veride kapasite artışı olduğu anlamına gelir. Uyarlanır yaklaşım ile karşılaştırıldığında Gzip, sıkıştırma aşamasında kapasite artışına sebep olmaktadadır.



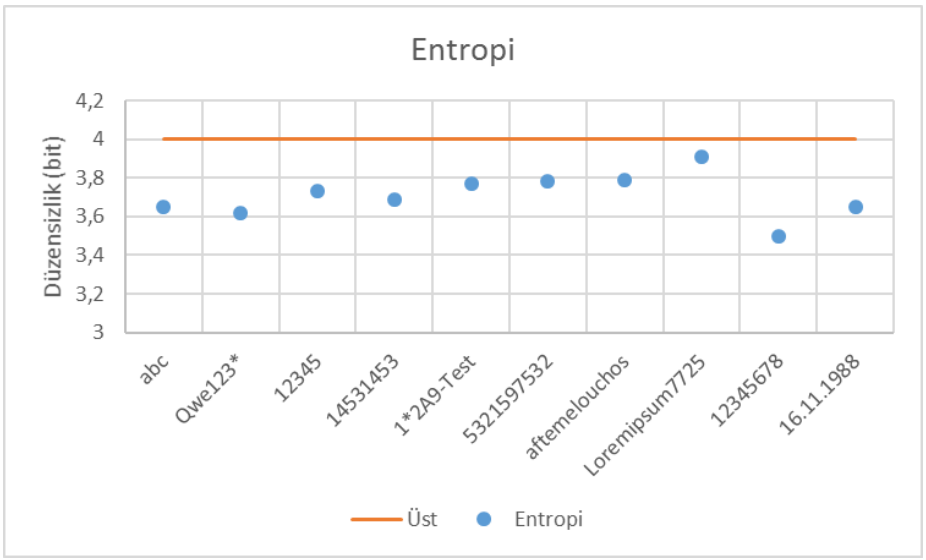
Şekil 7.4 Uyarlanır Huffman ve Gzip karşılaştırma.

AES şifreleme işleminde sıkıştırılan veri uzunluğuna bağlı olarak 16 bayt ve 32 baytlık harici veriler elde edilmektedir. Varsayılan bir ping paketinde bu değer 32 bayttır. Deneysel sonuçlar göz önünde bulundurulduğunda örneklerin %70'i varsayılan 32 baylık paket genişliğinin altında kalmaktadır (Şekil 7.5). Bu da etkin bant genişliği kullanımı olarak dikkat çekmektedir.



Şekil 7.5 Elde edilen paket boyutları.

Daha önceden belirtildiği gibi güvenli bir steganografik tasarının istatistiksel olarak algılanamaz olması arzu edilir. Deneysel sonuçlara ait entropi değerleri karşılaştırıldığında, tüm değerlerin 4'e yakın olduğu gözlenmiştir (Şekil 7.6). Entropi değerinin yüksek olması, orijinal mesaj (anahtar veri) ile gömülü mesaj arasında herhangi bir ilişkinin olmadığı anlamına gelir. Yapılan çalışmalar ve deneysel sonuçları neticesinde oluşturulan paket ile dağıtımı yapılacak olan anahtar veri arasında hiçbir ilgi olmadığı gözlenmiştir.



Şekil 7.6 Entropi değerleri.

8. SONUÇ VE ÖNERİLER

Bu çalışmada, kablosuz ağlarda anahtar dağıtımı için yeni bir protokol-içi ağ steganografi yöntemi önerilmiştir. Bu amaçla anahtar dağıtımı, süreç içerisinde yer alan fazlalık prosedürleri ortadan kaldırmak ve sistem dışı kullanıcılar için dikkat çekici olmaması koşulları dikkate alınarak ping komutu ile gerçekleştirilmiştir. Gizli veri olan anahtar bilgisi, ilk aşamada Uyarlanır Huffman kodlama algoritması kullanılarak sıkıştırılmış, daha sonra günümüzde veri şifreleme standartı olarak kullanılan AES ile şifrelenmiştir. Son olarak sıkıştırılıp şifrelenen anahtar veri paket içerisinde data alanına gizlenmiştir.

Geliştirilen bu tez çalışması kapsamı ve elde edilen sonuçlar neticesinde, konuya ilgi duyan araştırmacı ve bilim camiasına aşağıda yer alan öneri ve değerlendirmelerin sunulması uygun görülmüştür.

Açık haberleşme kanalları kullanılırken güçlü anahtar kullanılmasının yanı sıra kullanılan kriptolama yöntemi de oldukça önemlidir. Günümüzde kullanılan kriptolama yöntemleri güvenlik konusunda yeterli gözüküyorken, gelişen teknoloji ile birlikte bu yöntemlerin güvenliği tartışılmaya başlanacaktır. Bu durum, güvenliği önemsenen haberleşme sistemlerinde, kriptolamanın tek başına yeterli olmayacağı, başka yöntemlerle desteklenmesi gerektiği anlamına gelmektedir.

Birçok steganografik sistem, genellikle kaynak ve alıcının bu yöntemi bildiği düşünülerek tasarlanmaktadır. Kullanılan yöntemin sistem dışı kullanıcılar tarafından bilinme ihtimali de dikkate alınarak geliştirilen uygulama, şifreleme ile desteklenmiştir. Açık kanallar kullanılırken daha güvenli bir veri iletimi için kriptolama ile şifrelemenin bir arada kullanılması gerekmektedir.

Geliştirilen bir sistemden, etkin bant genişliği kullanımı, daha fazla bilgi depolama ve daha hızlı iletişim gibi özellikler bekleniyorsa, bu sistem içeresinde sıkıştırma teknolojilerinden yararlanmak gerekir. Ağ tabanlı geliştirilecek bir yöntemde, etkin bant genişliği kullanımı için sıkıştırma mutlaka olmalıdır. Yapılan çalışmanın kullandığı materyaller dikkate alınarak uygun sıkıştırma algoritmaları tercih edilmelidir. Metin tabanlı ve birbirini tekrar eden verilerin bulunduğu bir sistemde Uyarlanır Huffman kodlamanın kullanılması sıkıştırma performansını arttırmaktadır.

Kullanılan algoritma ve yöntemler, çalıştırıldıkları sistemin performansından etkilenirler. Gelişen teknoloji ile birlikte gerçekleşecek olan performans artışları, geliştirilen uygulamaların performanslarını olumlu yönde etkileyecektir.

Yapılan bu çalışma, sıkıştırma, şifreleme ve steganografi alanında sıklıkla kullanılan yöntem ve tekniklerin bir arada kullanılması ile gerçekleştirilmiştir. Böylelikle, önerilen metodun matematiksel karmaşıklığı arttırılmış ve girdi ile çıktı arasındaki doğrusallık azaltılmıştır.

9. KAYNAKÇA

[1] K. T. Chang C, "A reversible data hiding scheme using complementary embedding strategy", *Information Sciences,* no. 180, pp. 3045-3058, 2010.

[2] B. Elci, S. B. Örs ve V. Dalmışlı, "Bir Steganografi Sisteminin FPGA Üzerinde Gerceklenmesi", *3. Uluslararası Katılımlı Bilgi Güvenliği ve Kriptoloji Konferansı*, Ankara, 2008.

[3] A. Koltuksuz, "Elektronik Ticarette Güvenlik, Özgürlük Denetimi, Doğruluk, Bütünlük ve Sayısal İmza", *4.Türkiye İnternet Konferansı*, İstanbul, 1998.

[4] W. Stallings, Cryptography and Network Security: Principles and Practice, Prentice Hall, 1998.

[5] G. Galambos ve J. Bekesi, "Data Compression: Theory and Techniques", *Database and Data Communication Network Systems: Techniques and Applications*, USA, Elsevier Science, 2002, p. 233.

[6] M. Begum ve Y. Venkataramani, "LSB Based Audio Steganography Based On Text Compression", *Procedia Engineering,* cilt 30, pp. 703-710, 2012.

[7] H. Al-Bahadili, " A novel lossless data compression scheme based on the error correcting Hamming codes", *Computers & Mathematics with Applications,* cilt 56, no. 1, pp. 143-150, 2008.

[8] G. Pipeleers, J. Demeulenaere, J. Swevers ve L. Vandenberghe, "Extended LMI characterizations for stability and performance of linear systems", *Systems & Control Letters,* cilt 58, no. 7, pp. 510-518, 2009.

[9] L. Bei, W. Fen ve K. SungWan, "Switching LPV control of an F-16 aircraft via controller state reset", *IEEE Transactions on Control Systems Technology,* cilt 14, no. 2, pp. 267-277, 2006.

[10] J. Postel, "Internet protocol", RFC 791, 1981.

[11] L. Ji, W. Jiang, B. Dai ve X. Niu, "A novel covert channel based on length of messages", *Information Engineering and Electronic Commerce*, 2009.

[12] S. Zander, G. Armitage ve P. Branch, "An empirical evaluation of IP time to live covert", *15th IEEE International Conference on Networks*, 2007.

[13] O. Arkin ve J. Anderson, "EtherLeak: Ethernet Frame Padding Information Leakage", 2003. [Çevrimiçi]. Available: http://leetupload.com/database/Misc/Papers/atstake_etherleak_report.pdf. [Erişildi: 2016].

[14] B. Jankowski, W. Mazurczyk ve K. Szczypiorski, "PadSteg: introducing inter-protocol steganography", *K. Telecommun Syst.*, cilt 52, no. 2, pp. 1101-1111, 2013.

[15] C. H. Rowland, "Covert Channels in the TCP/IP Protocol Suite", *First Monday*, cilt 2, no. 5, 1997.

[16] K. Ahsan ve D. Kundur, "Practical data hiding in TCP/IP", *Proc. Workshop on Multimedia Security at ACM Multimedia*, 2002.

[17] A. Hintz, "Covert Channels in TCP and IP Headers", 2003.

[18] G. Fisk, M. Fisk, C. Papadopoulos ve J. Neil, "Eliminating steganography in Internet traffic with active wardens", *International Workshop on Information Hiding*, Berlin, 2002.

[19] W. Mazurczyk, M. Smolarczyk ve K. Szczypiorski, "Hiding information in retransmissions", *CoRR*, cilt abs/0905.0363, 2009.

[20] J. Postel, "Transmission Control Protocol", RFC 793, 1981.

[21] S. Cabuk, C. E. Brodley ve C. Shields, "IP covert timing channels: design and detection", *Proceedings of the 11th ACM conference on Computer and communications security*, Washington DC, 2004.

[22] R. Stewart, "Stream control transmission protocol", RFC 4960, 2007.

[23] W. Frączek, W. Mazurczyk ve K. Szczypiorski, "Hiding information in a Stream Control Transmission Protocol", *Computer Communications*, cilt 35, no. 2, pp. 159-169, 2012.

[24] L. Yao, X. Zi, L. Pan ve J. Li, "A study of on/off timing channel based on packet delay distribution", *Computers & Security*, cilt 28, no. 8, pp. 785-794, 2009.

[25] W. Mazurczyk, M. Smolarczyk ve K. Szczypiorski, "On information hiding in retransmissions", *Telecommun Syst*, cilt 52, pp. 1113-1121, 2013.

[26] X. Zou, H. Jin, K. Hao ve S. Sun, "Communication Hiding Algorithm Based on HTTP Protocol Parameters Sorting", *Computer Engineering*, cilt 32, no. 10, pp. 147-149, 2006.

[27] X. Zou, "Covert channels based on command sequence of FTP protocol", *Journal of Harbin Institute of Technology*, cilt 39, no. 3, pp. 121-126, 2007.

[28] K. Szczypiorski ve W. Mazurczyk, "Hiding data in OFDM symbols of IEEE 802.11 networks", *CoRR*, cilt abs/1006.0495, 2010.

[29] K. Szczypiorski, "A performance analysis of HICCUPS—a steganographic system for WLAN", *Telecommunication Systems*, cilt 49, no. 2, pp. 255-259, 2012.

[30] E. Tumoian ve A. Maxim, *Detecting NUSHU covert channels using neural networks*, aganrog State University of Radio Engineering, 2005.

[31] S. J. Murdoch ve S. Lewis, "Embedding covert channels into TCP/IP", *Proceedings of information hiding workshop*, 2005.

[32] B. Pfitzmann, "Information Hiding Terminology", *Information Hiding*, 978-3-540-49589-5, 1996, pp. 347-350.

[33] A. H. Murray ve R. W. Burchfield, The Oxford English Dictionary, U.K.: Oxford University Press, 1933.

[34] R. Chandramouli, M. Kharrazi ve N. Me, "Image Steganography and Steganalysis: Concepts and Practice", *Digital Watermarking*, U.S.A., Springer Berlin Heidelberg, 2004, pp. 35-49.

[35] B. Newman, Secrets of German Espionage, Right Book Club, 1940.

[36] I. J. Cox, M. L. Miller ve J. Bloom, "Watermarking Applications and Their Properties", *Int. Conf. on Information Technology*, Las Vegas, 2000.

[37] R. J. Anderson ve F. A. P. Petitcolas, "On the Limits of Steganography", *IEEE Journal of Selected Areas in Communications*, cilt 18, no. 4, pp. 474-481, 2008.

[38] N. F. Johnson ve T. Rude, "Introduction to Steganography: Hidden Information", *Regional Computer Forensic Group (RCFG) and Mid-Atlantic Chapter of High-Technology Crime Investigation Association (HTCIA) George Mason University Computer Forensics Symposium (GMU 2001)*, Fairfax, 2001.

[39] A. Şahin, E. Buluş ve M. Sakallı, "24-Bit Renkli Resimler Üzerinde En Önemsiz Bite Ekleme Yöntemini Kullanarak Bilgi Gizleme", *Trakya Üniversitesi Fen Bilimleri Dergisi*, 2006.

[40] A. Al-Nazer ve A. Gutub, "Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography", *NSS'09: Third International Conference on Network and System Security*, 2009.

[41] N. Samphaiboon, "Steganography via running short text messages", *Multimedia Tools and Applications*, cilt 52, no. 2, pp. 569-596, 2011.

[42] K. Alla ve R. S. R. Prasad, "An Evolution of Hindi Text Steganography", *Sixth International Conference on Information Technology: New Generations*, Las Vegas, 2009.

[43] N. F. Johnson ve S. Jajodia, "Stegananalysis of Images Created Using Current Steganography Software", *Second Information Hiding Workshop held in Portland*, Oregon, 1998.

[44] A. Westfeld ve A. Pfitzmann, "Attacks on Steganographic Systems", *Information Hiding*, Springer Berlin Heidelberg, 2000, pp. 61-76.

[45] D. Gruhl, A. Lu ve W. Bender, "Echo Hiding", *Proceeding of First International Workshop*, Cambridge, 1996.

[46] W. Bender, D. Gruhl, N. Morimoto ve A. Lu, “Techniques for data hiding”, *IBM Systems Journal*, cilt 35, 1996.

[47] Y. S. Kim, Y. M. Kim, J. Y. Choi ve D. K. Baik, “Information Hiding System StegoWaveK for Improving Capacity”, *Parallel and Distributed Processing and Applications*, Springer Berlin Heidelberg, 2003, pp. 423-436.

[48] B. Schneider, Applied Cryptography Second Edition, New York: John Wiley & Sons, Inc, 1996.

[49] E. Andiç, *Bilgisayar Haberleşmesinde Şifreleme (Kripto) Yazılımıyla Güvenliğin Sağlanması,* Marmara Üniversitesi, 2002.

[50] C. Narasimham ve J. Pradhan, “Evaluation of performance characteristics of cryptosystem using text files”, *Journal of Theoretical & Applied Information Technology*, cilt 4, no. 1, pp. 55-59, 2008.

[51] B. Gladman, “A Specification for Rijndael, the AES Algorithm”, 15 04 2003. [Çevrimiçi]. Available: http://techheap.packetizer.com/cryptography/encryption/spec.v36.pdf. [Erişildi: 2016].

[52] E. Şen, “Kriptografi ve Kullanım Alanları - I”, [Çevrimiçi]. Available: https://tr.scribd.com/doc/2581535/Kriptografi-ve-Kull-Alan-I. [Erişildi: 2016].

[53] R. Sedgewick, “Hashing”, *Algorithms in Java*, Addison-Wesley Professional, 2002.

[54] İ. Ciğer ve S. B. Yarman, *Data Şifreleme Algoritmaları ve Performans Analizi,* İstanbul Üniversitesi, 2012.

[55] M. T. Sakallı, *Modern şifreleme yöntemlerinin gücünün incelenmesi,* Trakya Üniversitesi, 2006.

[56] E. Biham ve A. Shamir, “Differential Cryptanalysis of DES-like Cryptosystems”, *Journal of Cryptology,*, cilt 4, no. 1, pp. 3-72, 1991.

[57] J. Daemen ve V. Rijmen, “AES Proposal: Rijndael”, *First Advanced Encryption Conferance*, California, 1998.

[58] S. Vaudenay, “An experiment on DES - statistical cryptanalysis”, *In Proceedings of Conference on Computer and Communications Security – CCS’95*, 1995.

[59] P. Junod ve S. Vaudenay, “Optimal Key Ranking Procedures in a Statistical”, *International Workshop on Fast Software Encryption*, 2003.

[60] C. Cid ve H. Gilbert, “AES Security Report : ECRYPT Report D.STVL.2”, *ECRYPT*, p. 73, 2006.

[61] J. Y. Liang, C. S. Chen, C. H. Huang ve L. Liu, “Lossless compression of medical images using Hilbert space-filling curves”, *Computerized Medical Imaging and Graphics*, cilt 32, no. 3, pp. 174-182, 2008.

[62] E. Uçar, *Sıkıştırılmış raster görüntülerin fotogrametrik otomasyonda kalite ve doğruluk üzerindeki etkilerinin araştırılması,* Selçuk Üniversitesi, 2011.

[63] B. Welch, "A technique for high-performance data compression", *Computer,* cilt 17, no. 6, pp. 8-19, 1984.

[64] A. Gersho ve R. M. Gray, Vector quantization and signal compression, New York: Springer US, 1992.

[65] R. O. Duda, P. E. Hart ve D. G. Stork, Pattern Classification 2nd Edition, U.S.A.: Wiley-Interscience, 2001.

[66] D. Salomon, A concise introduction to data compression, London: Springer-Verlag, 2008.

[67] M. Atalar, *İmge Dizilerindeki Artıkların İşlenmesi,* 2008.

[68] Y. Fisher, Fractal Image Compression, New York: Springer-Verlag , 1995.

[69] İ. Barış, M. Erdamar, E. Sümer ve H. Erdem, "Ses İşaretlerinin Yapay Sinir Ağları ile Tanınması ve Kontrol İşlemleri için Kullanılması", *URSİ,* 2002.

[70] D. Salomon, Data Compression, London: Springer-Verlag, 2007.

[71] S. Erman, *Kablosuz Ağlarda El Değiştirme,* İstanbul Teknik Üniversitesi, 2008.

[72] J. Wei-hua, L. Wei-hua ve D. Jun, "The application of ICMP protocol in network scanning", *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies,* 2003.

[73] C. H. Kim, "Differential Fault Analysis against AES-192 and AES-256 with Minimal Faults", *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2010.

[74] W. Mazurczyk, M. Karas ve K. Szczypiorski, "SkyDe: a Skype-based Steganographic Method", *CoRR,* cilt abs/1301.3632, 2013.

2. ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Çağatay AY

Doğum Tarihi ve Yeri : 16.11.1988, Yalova

Yabancı Dili : İngilizce

E-posta : cagatayay@duzce.edu.tr

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Eğitimi	Düzce Üniversitesi	2010
Lise	Bilgisayar / Yazılım	Aksa Anadolu Tek. Lisesi	2006