T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SOCKETLER ÜZERİNDEN ÖZEL HABERLEŞMEDE KRİPTOLOJİ METODLARIN KULLANILMASI VE BİR UYGULAMA

YÜKSEK LİSANS TEZİ

ORKHAN ABDULLAZADA

(Y1313.010038)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ahmed BABANLI

Şubat. 2017

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1313.010038** numaralı öğrencisi **Orkhan ABDULLAZADA**'nın **"SOCKETLER ÜZERİNDEN ÖZEL HABERLEŞMEDE KRİPTOLOJİ METODLARI KULLANILMASI VE BİR UYGULAMA"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 19.01.2017 tarih ve 2017/02 sayılı kararıyla oluşturulan jüri tarafından oybirliği ile Tezli Yüksek Lisans tezi olarak kabul edilmiştir.

Öğretim Üyesi Adı Soyadı	**İmzası**
Tez Savunma Tarihi :03/02/2017	
1)Tez Danışmanı: Prof. Dr. Ahmad BABANLI	.....
2) Jüri Üyesi : Yrd. Doç. Dr. Metin ZONTUL	.....
3) Jüri Üyesi : Yrd. Doç. Dr. Farzad KIANI	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum " Socketler Üzerinden Özel Haberleşmede Kriptoloji Metodların Kullanılması ve Bir Uygulama." adlı çalışmanın tezin proje safhasında sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığını ve yararlandığım eserlerin Bibliyografiya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim.(05.01.2017)

Orkhan ABDULLAZADA

ÖNSÖZ

Bu çalışmada beni yönlendiren ve bana yardımcı olan değerli hocam Prof. Dr Sayın Ahmed BABANLI'ya ve eğitimim boyunca emeyi geçen tüm hocalarıma teşekkür eder ve saygılarımı sunarım.

Şubat, 2017 **Orkhan ABDULLAZADA**

İÇİNDEKİLER

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

SOCKETLER ÜZERİNDEN ÖZEL HABERLEŞMEDE KRİPTOLOJİ METODLARIN KULLANILMASI VE BİR UYGULAMA

ÖZET

Teknolojinin gelişimi birçok insan faaliyetlerinin kolaylaştırılması avantajı ile beraberinde dezavantajlarıda sergilemiştir. Oluşan bu dezavantajlar, Teknoloji alanına yan dal olarak Teknolojinin güvenliği alanınıda geliştirilmesine sebep olmuştur. Teknoloji gelişim sürecinde gözden kaçırılan hata payları, algoritmik karmaşıklık değerleri ve birçok unsurlar dezavantaj olarak nitelendirilir. Her hangi sebepden oluşan dezavantajlar teknolojinin çalışma prensibini bozmakdan çok, kullanım alanında yanlış faaliyetleri yapabilitesini yükseltmektedir. Bu sebepden modern teknolojilerin yanlış faaliyet amaçlı kullanılmaması için bu teknolojileri koruyan teknolojiler geliştirilmiştir. Elektronik ortamlarda yapılan işlemlerin amaçlarını değiştirmemek adına yapılmış güvenlik sağlayıcı algoritmalardan biride şifreleme sistemleridir. Şifreleme sistemleri teknolojinin her dalına uygulanarak yapılacak işlemlerin dış güçler tarafından yönlendirememesini sağlamıştır. Teknoloji tarihi boyunca geliştirilmiş Şifreleme Sistemlerinin bazıları saldırganların önlemlerini yeterince alamaması nedeni ile ve ya cari Şifreleme sistemin yapılacak esas işlemin yavaşlatması nedeni ile teknolojinin bu dalı araştırma merkezlerinde dahada yüksek noktalara taşınmaktadır.

Şifreleme sistemleri aynı zamanda iletişimi sağlamak için geliştirilen teknolojiye uygulanarakda yapılacak iletişimin güvence altına almaktadır. Tez kapsamında yapılan araştırmada ilk şifreleme sistemlerinden, günümüz teknolojilerin kullandığı şifrelme sistemlerine kadar tümünün çalışma prensibi anlatılmaktaıdr. Tez kapsamında yapılan araştırmalara dayanılarak yapılan projede gizli anahtarlı şifreleme yöntemlerinden olan ve en son teknolojilerin kullandığı Rijndeal algoritması sayesinde ortadaki adam saldırısına uğranmasına rağmen iletişimin güvence altında yapılması sağlanmıştır. Bu projede iletişim esnasın ortadaki adam saldırısını gerçekleştiren saldırganın, yapılan iletişimin şifreli olduğuna dair şüphelerini kaldırılması için, aynı zamanda gizli anahtarlı şifreleme sisteminin doğası gereyi şifreleyici ve deşifreleyici anahtarın güvensiz kanal üzerinden de gönderile bilinmesi için Steganografi tekniği uygulanmıştır.

Anahtar Kelimeler: *Şifreleme sistemleri, Gizli Anahtarlı şifreleme, Açık Anahtarlı şifreleme, Steganografi, Socketler programlama.*

USİNG CRYPTOGRAPHİC METODS İN PRİVATE COMMUNİCATİON VİA SOCKETS AND AN APPLİCATİON

ABSTRACT

The advantage of Development of Technology are facilitated human activities. Along this advantages appeared disadvantages. These disadvantages are led to development Seceure Sistems of technology. The technology disadvantages are considered by overlooked errors, value of algorithm complexity. Any disadvantages are occured along side of distributing the working principle of technology, also misuse rate will be increase. For these reason protect to these technologies are being developed safe systems. One of the safe systems are the Criptpographic systems. Cryptography using every technology systems for secure.

Cryptography also using in electronic communication. Thanks to cryptography, pairs can communicate safely. In this Dessertation are reserched history of Cryptography, current Cryptography sistems which use in modern technology. Based on advanced Cryptographic sistems this dessertation project use Rijndeal algorithm and make this communication safely against atackers. In this project when pairs are during communication over the insecure channel, this channel are eavedropping from passive attacker. The purpose of this project is make commination safely. To make communication safely, pairs must communicate by Rijndeal cryptographic algorithm. Other purpose of this project is attacker shouldn't suspect in this communication using cryptographic techniques. For this reason pairs must hide the encrypted text by Steganographic techniques. İn this project pairs use private key encryption sistems, for this reason pair must share private key over the secure channel. İn this project pairs can share private key over the insecure channel by Steganographi techniques.

Keywords: *Encryption systems, Private key encryption, Public key encryption, Steganography, Socket programming*

1 GİRİŞ

Tez kapsamında yapılan araştırma, Kriptografi bilim dalının bölümleri hakkındadır. Bu araştırmada kriptografinin dallarında yapılmış teknikler anlatılmaktadır. Modern teknolojiler araştırmada gösterilen kriptografinin bütün dalını kullanmaktadır ki bu dallar işlevlerine göre uygun teknolojilere uygulanır. Tez kapsamında 2. Bölümde Kriptografi biliminin isminin nerden geldiği hakkında bilgi verilmiştir, alt bölümler olarak bu dallara değinilmiştir.

Kriptografi temel olarak 5 bölüme ayrılmaktadır.

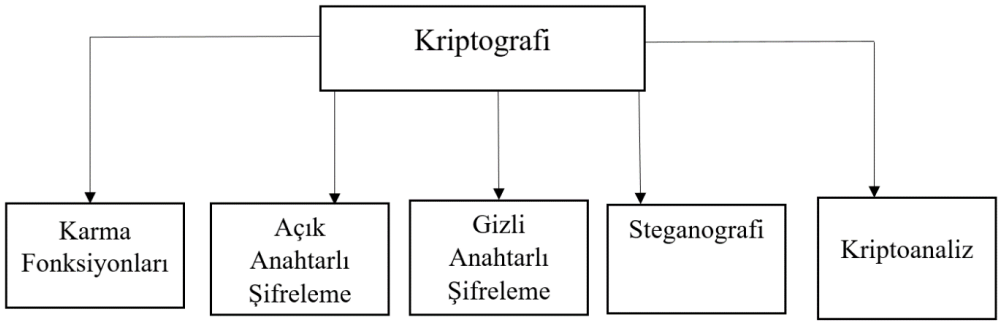
- Açık anahtarlı şifreleme sistemleri. Bu sistemlerin uygulanma amacı güvensiz kanal üzerinden iletişim sağlayacak tarafların iki tane anahtarı bulunmaktadır. Bu anahtarlardan bir tanesi her kes tarafından bilinen anahtardır ki karşı taraf orijinal mesajı şifrelerken herkes tarafından bilinen anahtarla şifrelemelidir. Bu şifreleme yönteminin önemli taraflarından bir tanesi ise deşifreleme işleminin sadece gizli anahtarla gerçeleştirilmesidir. Tez kapsamında açık anahtarlı şifreleme sistemlerine ait algoritmalar gösterilmiştir.
- Gizli anahtarlı şifreleme sistemleri. Bu sistemlerin uygulama amacı da güvensiz kanal üzerinden güvenli iletişimi sağlamaktır. Bu şifreleme sistemlerinde iletişime geçilecek tarafların ortak anahtarı belirlemeli ki bu anahtarı da güvenli kanal üzerinden iletişim yaparak belirlenmeli. Belirlenmiş olan gizli anahtar ile şifreleme ve deşifreleme işlemi yapılır. Tez kapsamında gizli anahtarlı şifreleme sistemine örnek olan algoritmalar incelenmiş, ve yapılan projede bu tür şifreleme sistemi kullanılmıştır.
- Steganografi, Kriptografinin esas bölümden biri olup yapılacak iletişimi şifrelemeden karşı tarafa gizli şekilde iletmek için kullanılmaktadır. Tez kapsamında yapılan araştırmada tarih boyunca ve modern teknolojide kullanılan steganografi teknikleri anlatılmış, ve yapılan uygulamada gizli anahtarlı şifreleme yöntemi ile iletişim sağlamak isteyen iki tarafın güvenli kanal olmaksızın iletişimi gerçekleşmesi sağlanmıştır.

- Karma Fonksiyonları Kriptografinin esas bölümlerindendir. Bu fonksiyonun çıktısı olan şifrelenmiş metinlerin deşifrelenmesi yoktur. Bu tür fonksiyonlar veri bütünlüğünün doğrulanması için kullanılmaktadır. Tez kapsamında yapılan araştırmada modern teknolojinin kullandığı Karma Fonksiyonları, bu fonksiyonların sağladığı avantajlar ve dezavantajlardan bahsedilmiştir.
- Kriptoanaliz, Kriptografinin bir başka esas bölümüdür ki, bu bölümde geliştirilmesi planlanan Şifreleme sistemlerininin tasarımı esnasında dikkat edilmesi gereken standatlar ele alınır. Tez kapsamında yapılan araştırmada kriptoanaliz standartlarına değinilerek geliştirilmek istenen Şifreleme Sisteminin dirençliliği nasıl denetlenmesi yapılmıştır.

Tez kapsamında 3. Bölümde iletişimin sağlanması için gereken bilgilere değinilmiştir ki bu bilgilerden yola çıkılarak 4. bölümde iki karakterin iletişimi sağlanmıştır. Yapılan iletişim ortadaki adam saldırısına uğramasına rağmen iletişim güvenli boyuta yükseltilmiştir.

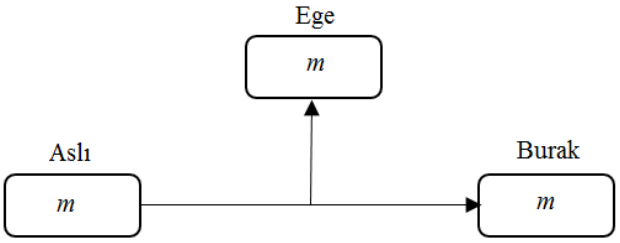
2 KRİPTOGRAFİ KAVRAMI

Kriptoloji bilimi güvenlik alanının en önemli dallarından biridir ve kelime kökeni yunancadan gelmektedir. Anlamı ise *Cryptos* gizlilik, *logos* bilim demektir. Kriptolojinin önemli dallarından biride Kriptografidir ki kelime kökeni yunancadan gelmektedir. Anlam olarak "*kryptos"* gizlilik , "*graphien"* yazmak ifade edilmektedir. Türkçe karşılık olarak *"Şifre yazımı"* diye de bilinmektedir. Kriptografi kendi bünyesinde beş ana dala bölünmektedir ki bu dalların şeması Şekil 2.1'da çizilmiştir.



Şekil 2.1: Kriptografi Şeması

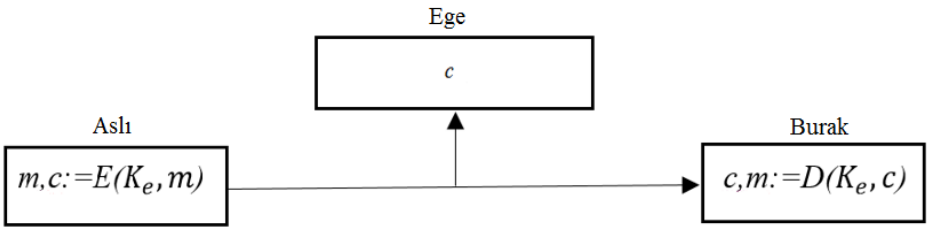
Kriptografinin amacı orjinal mesajı şifrelemek ve deşifrelemekten ibaretdir. Teknoloji dünyasının tüm alanlarında güvenlik ele alınması için yapılan tüm işlemler şifreleme ve deşifrelemeye dayanmaktadır. Bu güvenliyin esas amacı çok bilinen "*Ortadaki Adam*" saldırılarının önleminin alınmasıdır. Ortadaki adam saldırısı Şekil 2.2' de gösterilmiştir.



Şekil 2.2: Otadaki Adam Saldırısı

Şekil 2.2'den de görüldüğü gibi Aslı ve Burağın güvensiz kanal üzerinden sohbeti gerçekleştirilmiştir ve bu esnada Egenin ortadaki saldırgan şahıs olarak bu sohbeti dinlemektedir. Bu tür saldırıyı gerçekleştiren saldırganlar grup olarak ikiye ayrılırlar. *Aktiv-* saldırganlar güvensiz kanal üzerinden elde ettiği mesajları değiştire bilir, mesaj iletimini engelliye bilir, Mesajları iletme süresinin geciktire bilir ve ya mesaj sırasını değiştire bilir saldırganlardır. *Pasiv-* saldırganlar ise güvensis iletişim kanalını sadece dinleyen saldırganlardır.

Bu saldırganların önlemini alınması için Aslı ve Burak kendi aralarında mesajlaşırken Şifreleme ve Deşifreleme fonksiyonları kullanılmalıdır. Bu şekilde mesajlaşmalar ortadaki adam saldırganı olan Ege'i mesajı elde etmesinin önlemini almasada mesajın içeriğini öğrenmesinin önlemi alınacaktır. Şekil 2.3 de bu işlemler gösterilmiştir.



Şekil 2.3: Ortadaki adam saldırısının önlemi

Şekil 2.3'den de görüldüğü gibi Aslı mesajı göndermeden önce Burak ile ortak şifreleme ve deşifreleme anahtarını belirler K_e ki Egenin bu anahtarı bilmemesi gerekiyor. Ortak gizli anahtarı belirledikten sonra Aslı bu anahtar ile şifreleme ilemini gerçekleştirir $E(K_e,m)$, sonrasında şifrelenmiş olan mesajı c'i Burağa gönderir. Ege şifrelenmiş olan mesajı elde etse bile gizli anahtarı bilmediğine göre mesaj içeriğini göremeyecektir. Burak şifrelenmiş mesajı alır ortak belirlenen gizli anahtar ile deşifreleme fonksiyon sonucu orijinal mesajı elde eder. Aynı çözüm Burak içinde geçerli olucaktır

2.1 Gizli Anahtarlı şifreleme

Aslı ve Burağın güvenli iletişim kanalı üzerinden ortak belirlediği gizli anahtar ile şifreleme işleminin başka bir ifade ile simetrik şifreleme sistemi olarak da

bilinmektedir. Simetrik şifreleme işleminin anlamı her iki tarafın aynı anahtarı kullanarak aynı şifreleme ve deşifreleme fonksiyonlarının kullanmasıdır.

İlk gizli anahtarlı şifreleme fonksiyonu Roma imparatoru "Julio Ceaser"-ın keşfettiği Sezar şifreleme yöntemidir. Savaş esnasında Generallerle mektuplaşarak savaş hakkında bilgileri ve vereceği emirleri düşman askerler tarafından ele geçirilse bile okunamaz hale getirmek için kullandığı yöntemdir. Sezar şifreleme sisteminin çalışma prensibi ise Generaller ve sadece kendisinin belirlediği anahtar numarası seçilir ki bu anahtar numarası alphabe sırasını kaydırmak için kullanılır. Çizelge 2.1 da gösterildiği gibi, eğer gizli anahtar *K=1* olarak ele alacak olursak, şifreleme işlemini kaydırılmış apfabeye göre yapıla bilinir.

Çizelge 2.1: Sezar Şifreleme Çizelgesu

Afabe	**A**	**B**	**C**	**Ç**	**D**	**E**	**F**	**G**	**Ğ**	**H**	**I**	**İ**	**J**	**K**	**L**	**M**	**O**	**N**	**Ö**	**P**	**R**	**S**	**Ş**	**T**	**U**	**Ü**	**V**	**Y**	**Z**
Kaydırılmış Alfabe	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	O	N	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A

Çizelge 1.0'a göre şifrelemek isteyeceğimiz olan "AYDIN" kelimesinin şifrelenmiş hali "BZEİÖ" gibidir. Formül olarak anlatıcak olursak şifreleme işleminin formülü *harfin sıranumarası+1= uygun olan harif* deşifreleme işleminin formülü ise *şifreli metnin harf sırası-1=orıijinal metnin harf sırası.*

Gizli anahtar şifreleme kendi bünyesinde ikiye ayrılırlar ki bunlar

1. Bireysel Akış Şifreleme sistemi
2. Blok şifreleme sistemi

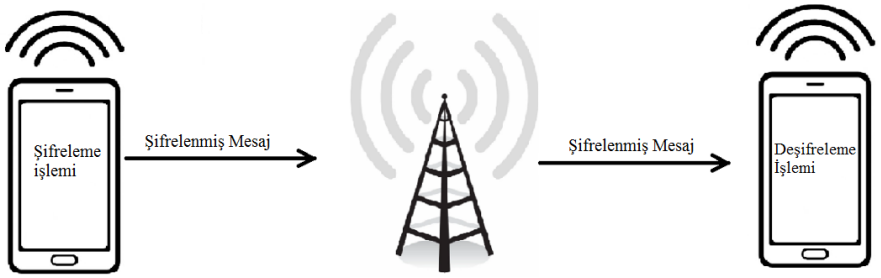
Bu şifreleme sistemlerinin her ikisi günümüz teknolojisinde kullanılmaktadır ki bunlardan bir kaçı saldırıya uğrayarak kırılmıştır.

2.1.1 Bireysel Akış Şifrelemesi

Simetrik Şifreleme Sisteminin önemli bölümlerinden olan Bireysel Akış Şifreleme Sistemi ile Blok Şifreleme sistemini karşılaştırılıcak olursak, bu şifreleme sistemi Blok şifreleme sistemine nazaran daha hızlı çalışmakta ve aynı zamanda daha az donanım maliyyetine sahiptir.

Adından da bilindiği üzere Bireysel Akış şifrelemesi orijinal mesajı şifrelerken herbir karakteri ayrı ayrı şifrelemektedir. Başka bir deyişle her bir karakter seçilmiş olan gizli anahtar ile bireysel olarak şifrelenir.

Modern teknolojinin en ünlüsü olan GSM (Global System for Mobile- Mobil İletişim İçin Küresel Sistem) Bireysel Akış Şifreleme sistemini kullanmaktadır. GSM çalışma prensibi iletilecek ses mesaj ikili sisteme çevrilerek bireysel olarak şifrelenir sonrasında baz istasyonuna gönderilir ve sonra bu mesaj hedef telefonuna gönderilir ve deşifreleme işlemi hedef telefonda gerçekleştirilir. Bu prensip Şekil 2.4 gösterilmiştir.



Şekil 2.4: GSM çalışma prensibi

Bireysel Akış Şifrelemesinin en önemli özelliklerinden birisi de zamana göre farklı şifreleme işlemini gerçekleştirmektir, ve bu yöntemle çalışan Şifreleme sistemlerinden en çok bilinen "Vernam Şifreleme Sistem" gösterile bilinir.

Vernam Şifreleme sistemi kendi ismini bu yöntemi keşfeden Gilbert Vernam' dan almıştır. Bu yöntem "Tek Kullanımlık Şerit" olarakda bilinir. Bu şifreleme sistemine göre her kelimeyle eş uzunlukta anahtar üretimi yapılmaktadır. Örnek olarak Türk alfabesine göre "AYDIN" kelimesini Vernam şifreleme yöntemi ile şifreleyelim. Bu yönteme göre her gönderilen mesaj için ayrı anahtar üretilir ki bu da kullanılan alfabedeki harf sınırları içerisinde farklı permutasyonlarda numaralar üretilir.

Çizelge 2.2: Vernam şifreleme Anahtar üretimi

Alfabe	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
Sıra numarası	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Rastgele üretilmiş Anahtar	23	12	9	6	11	22	18	13	10	7	5	16	24	15	8	1	26	3	20	4	21	2	25	19	17	14	29	27	28

Çizelge 2.2'de görüldüğü üzere "AYDIN" kelimesi için anahtar üretilmiştir. Vernam şifreleme sistemine göre şifrelenecek metnin içindeki harflerin sıra

numarasıyla onlara karşılık üretilmiş rastgele anahtar numarası toplanarak çıkan sonuç alfabe sayısının numarası ile modüler işleme tabi tutulur. “A” sıra numarası “1” ve karşılık gelen rastgele anahtar numarası “23”, bu rakamlar toplanarak mod 29 göre hesaplanıcaktır

$$1+23=24 \Leftrightarrow 24 \bmod 29=24 \text{ (T)}$$

$$28+12=40 \Leftrightarrow 40 \bmod 29=11 \text{ (I)}$$

$$5+9=14 \Leftrightarrow 14 \bmod 29=14 \text{ (K)}$$

$$11+6=17 \Leftrightarrow 17 \bmod 29=17 \text{ (N)}$$

$$17+11=28 \Leftrightarrow 28 \bmod 29=28 \text{ (Y)}$$

Böylece “AYDIN” kelimesinin Şifrelenmiş hali “TIKNY” olucaktır. Şifrelenmiş mesajı karşı tarafa göndermeden önce rastgele üretilmiş anahtar kelime güvnilir kanal üzerinden gönderilir sonrasında şifrelenmiş mesaj gönderilir. Gönderilen mesajı şifrelemek için, şifreli mesaj karakterlerinin alfabedeki sıra numarası ele alınır ve rastgele üretilmiş anahtar sırasını sırasıyla çıkarma işlemi yapılarak orijinal metin ele alınmış olur. “TIKNY” kelimesini deşifrelemek için bu karakterlerin alfabedeki sıra numarası belirlenir “24, 11, 14, 17, 28”, sonra bu rakamlardan rastgele üretilmiş olan anahtar dizisindeki birinci elemandan itibaren çıkarma işlemi gerçekleştirilir.

$$24 – 23=1 \text{ (A)}$$

$$11 – 12=-1+29=28 \text{ (Y)}$$

$$14 – 9=5 \text{ (D)}$$

$$17 - 6 = 11 \text{ (I)}$$

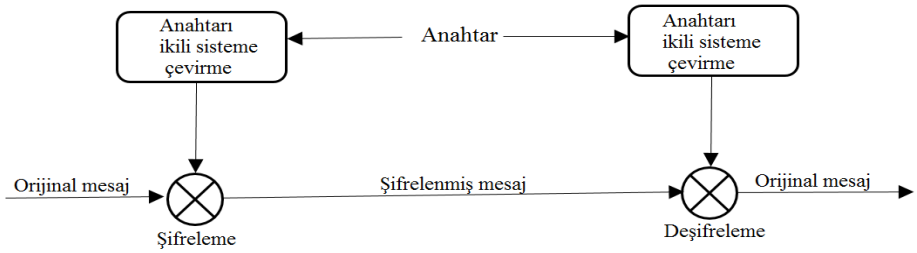
$$28 – 11 =17 \text{ (N)}$$

Böylece şifrelenmiş mesajı deşifrelenir. Günümüz teknolojisinde ise Vernam şifreleme sistemi orijinal messaj ve rastgele üretilmiş anahtar kelime ikili sisteme çevirilerek uygun gelen anahtar ile XOR işlemine tabi tutulur ve üretilmiş olan ikili sistem onlu sisteme çevrilerek şifrelenmiş mesaj elde edilir. Deşifreleme işlemi ise yine aynı şekild şifrelenmiş mesajın karakterleri ve belirlenmiş gizli anahtar ikili sisteme çevrilerek XOR işlemine tabi tutularak orijinal mesajı elde edilmiş olunuyor.

Bireysel Akış şifreleme sistemi temel olarak iki gruba bölünmektedir.

1. Eşzamanlı Bireysel Akış Şifreleme
2. Oto Eşzamanlı Bireysel Akış Şifreleme

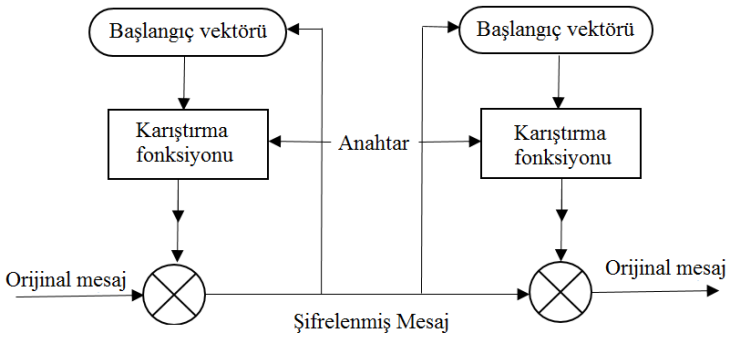
Eşzamanlı bireysel Akış Şifreleme sistemlerinde üretilen anahtar orijinal mesajdan ve ya şifrelenmiş mesajdan bağımsız olarak üretilir. Bu tür sistemlerde güvenli kanal üzerinden anahtarlar ikili sisteme çevrilerek gönderilmektedir, eğer anahtar gönderme zamanında veri kaybı yaşanırsa Bireysel Akış şifreleme sisteminin çalışma prensibi bireysel olduğu için bu bozulma diğer mesajlara yansımyacaktır. Aktiv saldırganlar şifrelenmiş mesajı ele geçirib karakterleri değiştirerek karşı tarafa gönderirse alıcı taraf gizli anahtarla bile şifrelenmiş mesajı deşifreleyemeyecektir. Dolayısıyla Eşzamanlı Bireysel Akış Şifreleme sistemleri Aktiv saldırılara karşı hassas hale gelmektedir. Bu işlemler Şekil 2.5 de gösterilmiştir



Şekil 2.5: Eşzamanlı Bireysel Akış şifreleme diagramı

Şekil 2.5'den görüldüğü üzere gizli anahtar bağımsız olarak ikili sisteme çevrilerek orijinal mesajın ikili sisteme çevrilmiş hali ile XOR işlemine girer.

Oto Eşzamanlı Bireysel Akış Şifreleme sistemlerinde gizli anahtarın üretimi sonrası, gönderilmiş şifrelenmiş mesaja göre üretilir ve karşı tarafa gönderilir. Şekil 2.6 da Oto Eşzamanlı Bireysel Akış Şifreleme sisteminin diagramı çizilmiştir.



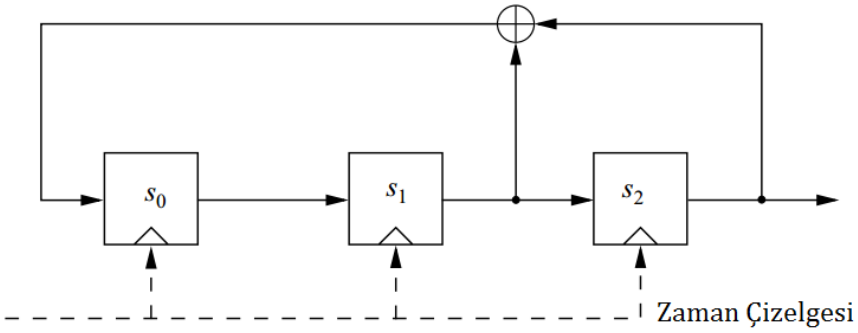
Şekil 2.6: Oto Eşzamanlı Bireysel Akış Şifreleme sistemi diagramı

Şekil 2.6'dan da görüldüğü üzere şifreleme başlarken bit olarak rastgele bir başlangıç vektoru ile ikili sisteme çevrilmiş gizlii ahatar karıştırma fonksiyonuna girer ve bu fonskiyonun çıktısı orijinal mesajın ikili sisteme çevrilmiş hali ile XOR işlemine tabi tutulur. Bir sonraki adımda ise önceki adımın çıktısı olan şifrelenmiş metin başlangıç vektörü rolünü oynar ki sonrasında bu vektör karıştırma fonksiyonu sayesinde gizli anahtar ile karıştırılır. Bu fonksiyonun çıktısı ikili sisteme çevrilmiş orijinal mesaj ile XOR işlemine tabi tutulur.

2.1.1.1 Bireysel Akış Şifrelemesi Anahtar Üretimi

Güvenli iletişimin Bireysel Akış Şifreleme yöntemi ile sağlanmasında farklı yöntemlerden biride her gönderilen şifrelenmiş mesajın deşifrelenmesi için farklı anahtarın üretilmesidir. Farklı anahtarın üretimi belirlenmiş doğrusal hareket formüllerine göre tasarlanır. Bu tür tasarlanan formüllerin genel adı ise Doğrusal Geribildirim Kaydırma Yazmaçları (LFSR-Linear Feedback Shift Register) olarak bilinmektedir. DGKY'ler teknikleri ile anahtarın üretimi donanım tarafından kolaylıkla gerçeklenebilinir. Matematiksel olarak kolay şekilde hesaplanabilinir.

DGKY'ler'n en basitleştirilmiş olanlarından birisi zamanın deyişmesinden bağımlı şekilde doğrusal olarak üretilen anahtarlardır. Şekil 2.7'de Basitleştirilmiş DGKY'nin örneği gösterilmiştir.



Şekil 2.7: Basitleştirilmiş DGKY diagramı

Şekil 2.2 diagramından da görüldüğü gibi bu DGKY'nin derecesi 3 olması gizli anahtarın ikili sistemdeki bit sayısının 3 olmasına dayanır. Diagramdan görüldüğü üzere her saniye geçtikce anahtar bitleri sağa doru kaydırılarak farklı anahtar üretiliyor. Bu üretim belirli bir zaman sonra kendini tekrarlayarak devam edecektir. Örnekte verilen s_0, s_1, s_2 değerleri gizli anahtarın bitlerini ifade etmektedir. Bu üretim belli bir zaman aralığında bitler her defasında sağa kaydırılarak ve bundan başka en sağdaki iki bit kendi aralarında XOR işlemine tabi tutularak kaydırılır. Gizli anahtarın "1 0 1" olduğunu varsayarsak Çizelge 2.2 de bu anahtarın farklı 8 farklı permutasyonları gösterilmiştir.

Çizelge 2.2: Gizli Anahtarın 8 farklı permutasyonu

Zaman	s_0	s_1	s_2
0	1	0	1
1	1	1	0
2	0	1	1

Çizelge 2.2'den de görüldüğü gibi belirlenmiş zaman aralığında farklı permutasyon üretilmişdir. 3. zaman aralığını, verilen diagrama göre hesaplayacak olursak 2. zamanda gizli anahtarın permutasyonu $s_0 = 0$, $s_1 = 1$, $s_2 = 1$ şeklindedir. Verilen diagrama göre işlem $s_1 = s_0$, $s_2 = s_1, s_0 = s_1 \otimes s_2$ şeklinde ilerlerse $s_1 = 0$, $s_2 = 1$, $s_0 = 0 \otimes 1 = 1$ Başlangıç durum anahtarını tekrarlayacaktır. Verilen bu örnek DGKY'nin en basitleştirilmiş şeklidir ki günümüz teknolojileri belirlenmiş zaman aralığında polinomik denklem hesaplamaları yapılarak gizli anahtar üretimini yapmaktadır.

2.1.2 Blok Şifreleme

Simetrik şifreleme sisteminin bir diğer alt bölümü de blok şifrelemedir. Blok şifrelemenin çalışma presnibi girilmiş olan orijinal mesajı bloklar haline getirerek şifreler ve şifreleme işleminin sonunda bu blokları yan yana dizerek şifrelenmiş mesajı üretmiş oluyor.

Blok şifreleme sisteminin en eski örnek olarak Vigener şifreleme sistemini göstere biliriz. Vigener şifreleme sistemine göre şifrelenecek kelime anahtar uzunluğu boyutunda bloklara bölünerek şifrelenir. Örnek olarak "Deformasyon" kelimesini "su" anahtar kelimesine göre şifreleyecek olursak kelime ikişer blok haline getirlir "De", "fo", "rm", "as", "yo", "nd" olarak bloklar haline getirilir, sonra bu bloklardaki harflerin alfabedeki sıra numarasıyla gizli anahtar içerisinde bulunan harflerin alfabedeki sıra numarası toplanır ve sonucunda mesajın şifrelenmesi gerçekleşiyor.

Modern blok şifreleme dikkat edilmesi gereken noktalar gözden kaçınılırsa "Kaba kuvvet (Brute force)" ve benzeri saldırılara karşı zayıf kalacaktır. Bu zayıflığı aradan kaldırma nedeni ile Blok şifrelemeleri güçlü kılan faktörler aşağıdakilerdir.

- Anahtar: Blok şifreleme sistemlerinde seçilen gizli anahtarın uzunluğunun artması saldırganların işlerinin bir o kadar da zorlaştırıldığı görülmüştür.
- Döngü sayısı: Blok şifreleme sistemlerinde yapılan döngülerin artırılması yerdeyişme fonksiyonlarının sayısını da arttırır, dolayısıyla şifreleme işlemi sonucu ortaya yeteri kadar karmaşık şifrelenmiş mesaj üretilmiş olur. Dikkat edilmesi gereken nokta ise döngü sayısının artışı algoritmanın karmaşıklığınıda artırmaktadır ki bunun sayesinde algoritmanın performansını etkilemiş olur
- S-kutuları (Substitution Box – Yerdeyişme kutuları): Blok şifreleme sistemlerinde orijinal mesajı karıştırma için S- kutularının rolü çok önemlidir. Günümüz teknolojisinin kullandığı şifreleme sistemlerinde S-Kutuları kullanılmıştır. S-kutuları Blok şifreleme sisteminin doğrusal olmayan elemanlarıdır ve bu yüzden iyi bir S-kutuların seçimi şifrenin karmaşıklığını doğrudan etkileyecektir.

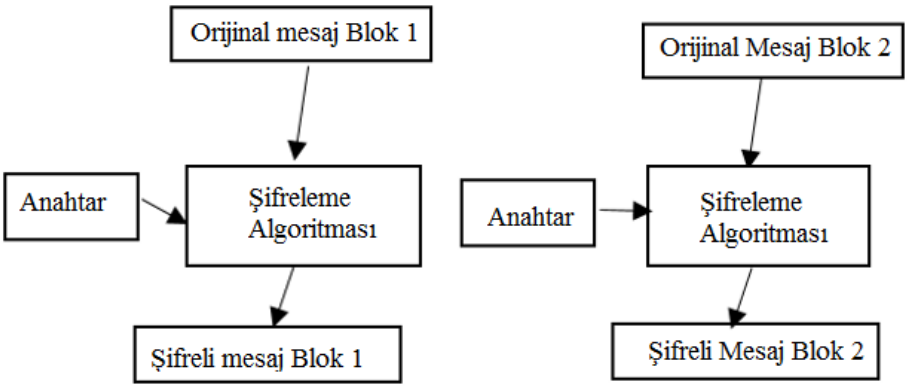
Modern Blok şifreleme sistemler yukarıda söylediğimizler döngüleri kendi içinde barındırmaktadır. Bu döngüler içerisinde Blok şifrelemelerin ait olan 5 metodlar bulunmaktadır bir çok blok şifreleme sistemleri bu metodlar prensibinde çalışmaktadır

- Elektonik Kod Defteri (Elecrtonic Codebook Mode – ECB-)
- Şifre Blok Zincirlemesi (Cipher Block Chaining Mode - CBC)
- Yayımlı Şifre Blok Zincirlemesi (Propagating Cipher Block Chaining Mode - PCBC)
- Şifre Geri Beslemeli (Cipher Feedback Mode - CFB)
- Çıktı geri Beslemeli(Output Feedback Mode-OFB)

Modern şifreleme algoritmaları gösterilen her moda göre farklı şifrelenmiş mesaj çıkartmaktadır.

2.1.2.1 Elektronik Kod Defteri Modu

Elektonik Kod Defteri Modu blok şifreleme sisteminin en basitlerindendir yukarıda gösterilen Veginer şifreleme sistemi çalışma prensibi bu metodla aynıdır. Şöyle ki şifrelenecek olan açık mesaj ikili sisteme çevrilerek bloklara bölünür ve her bir blok ayrı ayrı şifrelenir. İşlem sonucunda ise şifreli mesaj üretilmiş oluyor. Şekil 2.8'de Elektonik Kod Defteri modunun diagramı gösterilmiştir.

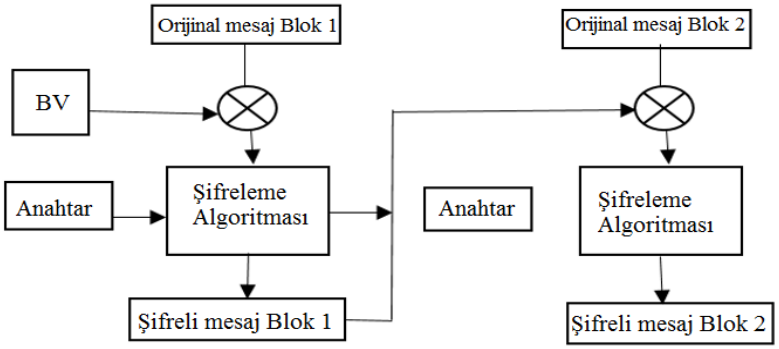


Şekil 2.8: Elektonik Kod Defteri diagramı

2.1.2.2 Şifre Blok Zincirlemesi Modu

Şifre Blok Zincirlemesi metodunda ikili sisteme çevrilmiş olan orijinal mesajın birinci bloğu Başlangıç Vektoru ile XOR işlemine tabi tutulduktan sonraki çıktı gizli anahtar ile şifreleme fonksiyonundan geçerek birinci şifrelenmiş bloğu

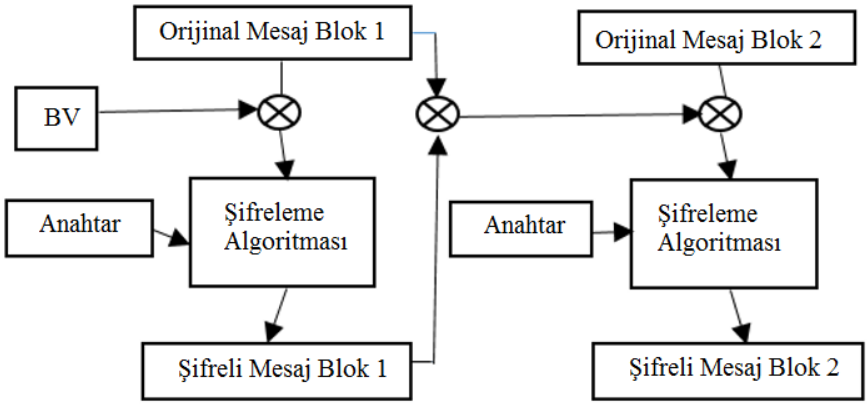
üretilir (Başlangıç Vektorü rastgele üretilmiş ve ya belirli bir algoritma sonucu olan ikili sistem çıktısı ola bilir). Bir sonra ki orijinal mesaj bloğunu şifrelemek için birinci şifrelenmiş olan mesajların bit karşılığı ile XOR işlemine tabi tutulduktan sonra şifreleme fonksiyonuna girilir. Şekil 2.9'de Şifre Blok Zincirlemesi modunun diagramı gösterilmiştir.



Şekil 2.9: Şifre Blok Zincirlemesi metodunun diagramı

2.1.2.3 Yayımlı Şifre Blok Zincirlemesi Modu

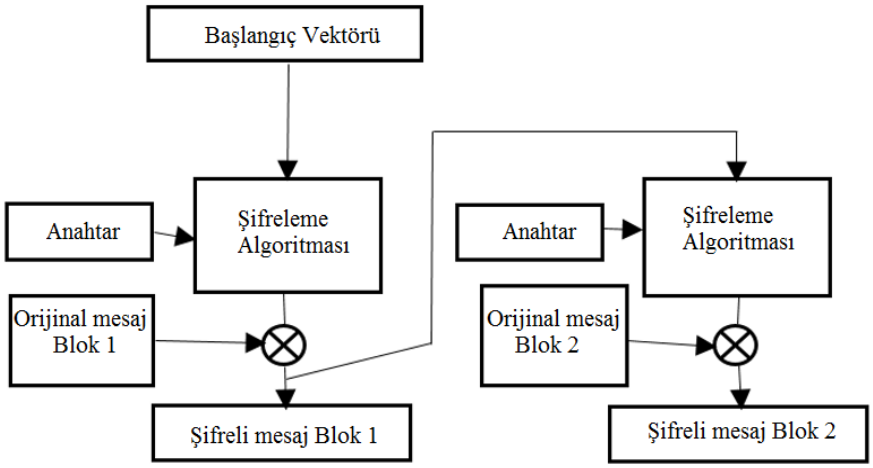
Yayımlı Şifre Blok Zincirlemesi metodu ise birinci ikili sisteme çevrilmiş orijinal mesaj bloğu Başlangıç vektörü ile XOR işlemine girer bu işlemin çıktısı ile gizli anahta şifreleme algoritmasına girer. Bu algoritmanın çıktısı birinci şifreli mesaj bloğunu oluşturur. ikinci bloğun şifrelemesinden önce birinci bloğun şifreli mesajı ile orijinal mesajın ikili sisteme çevrilmiş hali XOR işlemine girer, bu işlemin çıktısı ikinci şifrelenecek olan orijinal mesajın ikili sisteme çevrilmiş hali ile XOR işlemine girer. Bu işlemin çıktısı ile gizli anahtar şifreleme algoritmasına girer ve çıktı olarak ikinci şifrelenmiş mesaj bloğu üretilmiş olur. Şekil 2.10'da Yayımlı Şifre Blok Zincirlemesi modu diagramı çizilmiştir.



Şekil 2.10: Yayımlı Şifre Blok Zincirlemesi metodu diagramı

2.1.2.4 Şifre Geri Beslemeli Modu

Şifre Geri Beslemeli metodunda birinci şifreli mesaj bloğunu üretmek için rastgele seçilmiş Başlangıç metodu gizli anahtar ile Şifreleme algoritmasına girer ve bu algoritmanın çıktısı ile ikili sisteme çevrilmiş birinci orikinal mesaj bloğu XOR işlemine girerek birinci şifrelenmiş mesaj bloğunu üretmiş olur. Bir sonraki şifreli mesaj bloğunu üretmek için birinci şifreli mesaj bloğu gizli anahtar ile şifreleme algoritmasından geçer bu algoritmanın ikili sistem çıktısı ile ikinci ikili sisteme çevrilmiş orijinal mesaj bloğu ile XOR işlemine tabi tutulur ve çıktı olarak ikinci şifreli mesaj bloğu üretilmiş olur. Şekil 2.11'de Şifre Geri Beslemeli metodunun diagramı çizilmiştir.

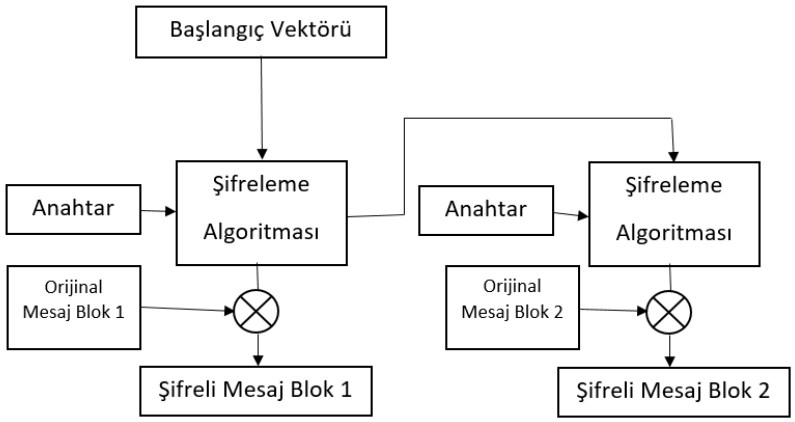


Şekil 2.11: Şifre Geri Beslemeli metodunun diagramı

2.1.2.5 Çıktı Geri Beslemeli Modu

Çıktı geri Beslemeli Metodunda Başlangıç metodu ile gizli anahtar şifreleme algoritmasına girer ve bu algoritmanın çıktısı hem ikili sisteme çevrilmiş

orijinal mesah blok 1 ile XOR işlemine girerek şifrelenmiş mesaj bloğunu üretir hemde ikinci şifrelenmiş mesaj bloğunu üretmek için gizli anahtar ile şifreleme algoritmasına girer. Bu algoritmanın çıktısı ikinci ikili sisteme çevrilmiş olan orijinal mesaj ile XOR işlemine girerek ikinci şifrelenmiş mesaj bloğunu üretir. Şekil 2.12'de Çıktı geri Beslemeli Metodunun diagramı çizilmiştir.

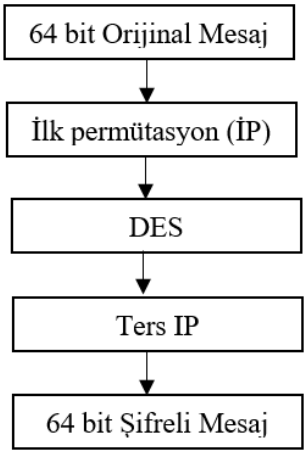


Şekil 2.12: Çıktı geri Beslemeli Metodunun diagramı

2.1.2.6 Veri Şifreleme Standardı (Data Encription Standart)

Ulusal Standartlar ve Teknoloji Enstitüsü (National Institude of Standards and Technology) 1977 senesinde Teknolojinin güvenliği arttırma amacı ile ulusal bir standart olabilecek kriptografik bir algoritma talebinde bulundu. Bu talebe yönelik IBM firmasının geliştirdiği LUCIFER algoritmasını Ulusal Standartlar ve Teknoloji Enstitüsüne sunmuştur. Bu algoritma üzerinde değişiklikler yapılarak bu algoritmanın resmi adı DES (Data Encription Standart) olarak belirlenmiştir.

DES algoritması bir Blok şifreleme sistemidir ki her bir bloğun boyutu 64 bitdir. Şekil 2.13'de DES algoritmasının genel yapısı gösterilmiştir.



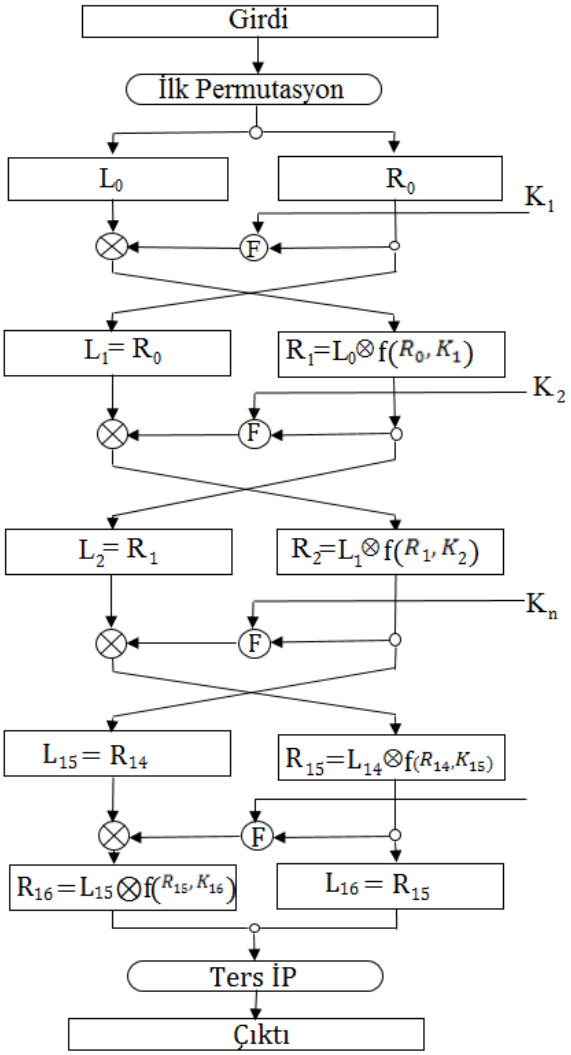
Şekil 2.13: DES algoritmasının genel yapısı

Şekil 2.13'de gösterildiği gibi şifrelenecek mesaj 64 bitlik bloklara bölünür ve her bir blok Şekil 2.14'de gösterilen İlk permutasyon işleminden geçer ki bu işlemin sonunda orijinal ilgili 64 bit orijinal mesajın bit sırası rastgele karıştırılıyor.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

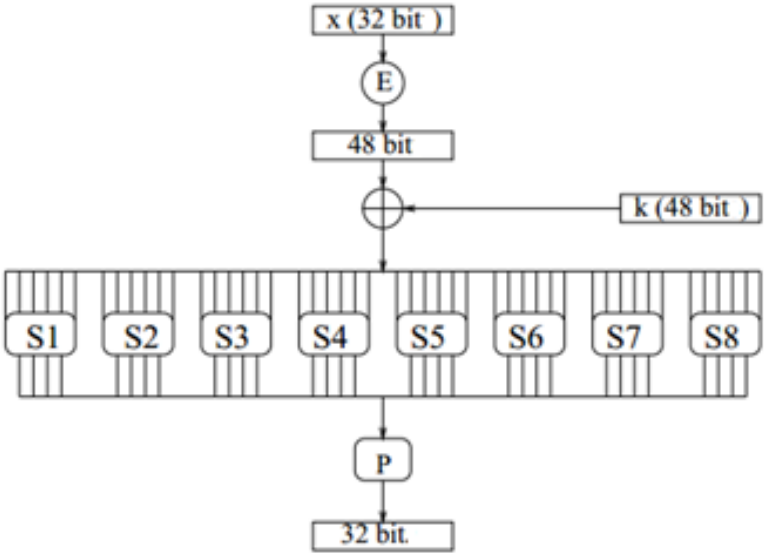
Şekil 2.14: İlk permutasyon işlemi

Şekil 2.14'de gösterildiği gibi ilgili 64 bitlik orijinal mesaj bu işlem sonucun de 58. Bit ilk 50. Bit ikinci 7. Bit ise 64 cü sırada durmuş olucaktır. Bu işlemin sonucu DES algoritmasının girdisi olarak Şekil 2.15' de gösterilen işlemlerden geçer



Şekil 2.15: DES diagramı

DES'in diagramından görüldüğü gibi 64 bitlik olan orijinal metin ilk permutasyon işleminden geçerek bitlerin sırası rastgele olarak dizilir, bu işlemin sonucundaki 64 bitlik veri sağ ve sol taraf olarak ikiye bölünür. Sağ tarafdaki 32 bitlik veri ile anahtar üretimi algoritmasının birinci turunun çıktısı F fonksiyonuna girer ki bu işlemin sonucunda 32 bitlik bir karıştırılmış veri üretilmiş olur. Şekil 2.16'da F fonksiyonunun diagramı gösterilmiştir.



Şekil 2.16: F fonksiyonu diagramı

F fonsiyonun diagramından görüldüğü gibi 32 bitlik sağ taraf genişletme fonksiyonuna girerek 48 bite yükseltilir. Genişletme fonksiyonu Şekil 2.17 gösterilmiştir.

32	31	30	29	28	27	26	25
24	23	22	21	20	19	18	17
16	15	14	13	12	11	10	9
8	7	6	5	4	3	2	1

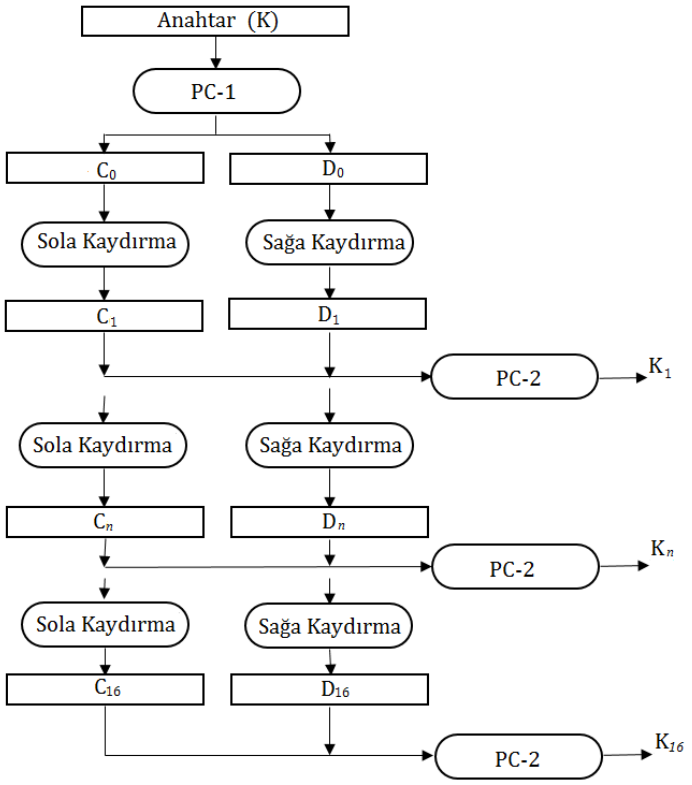
4	3	2	1	32	31	30	29	28	27	26	25
28	27	26	25	24	23	22	21	20	19	18	17
20	19	18	17	16	15	14	13	12	11	10	9
12	11	10	9	8	7	6	5	4	3	2	1

Şekil 2.17: Genişletme fonksiyonu

Genişletme fonksiyonunun resminden de görüldüğü gibi genişletme işlemi rastgele seçilen bitler tekrar olarak farklı sıraya yazılarak 32 bitlik veri 48 bitlik veri haline gelir.

Genişletme fonksiyonunun 48 bitlik çıktısı ile anahtar üretimi algoritmasının birinci turunun 48 bitlik çıktısı XOR işlemine tabi tutulur. Bu işlemin 48 bitlik sonucu sekiz parçaya bölünerek S kutularına girer.S kutularına giren 6 bit sayısı

4 bit olarak çıkar bu işlem ise S[0]={x_0, $\underbrace{x_1, x_2x_3x_4}$ x_5}→{y_0,y_1,y_2,y_3,} gerçekleştirilir. Bütün S kutularının sonucunda tüm 4 bitlik veriler birleşerek 32 bitlik bir veri haline gelir. F fonksiyonunun 32 bitlik çıktısı ile sol tarafdaki 32 bitlik veri ile XOR işlemine tabi tutulur. Bu işlemin çıktısı bir sonraki tur için sağ taraf rolünü oynuyacaktır. . Birinci turun sağ tarafı ise bir sonraki tur için sol tarafı olacaktır. On altıncı turdan sonra 64 bitlik çıktı Ters IP işemine girerek bitlerin sırası değiştirilir. Böylece her bir 64 bitlik veri için yukarıda anlatılan şifreleme işlemi 16 kere tekrarlanarak şifreleme işlemi gerçekleştirilir. Şifreleme işlemi gerçekleşitirilirke her bir tur için farklı bir anahtar üretilir. DES algoritmasının anahtar üretimi Şekil 2.18 de gösterilmiştir.



Şekil 2.18: DES algoritması Anahtar üretimi diagramı

DES algoritması Anahtar üretimi diagramından da görüldüğü üzere anahtar üretimi işlemi 16 turdan ibaretdir. DES algoritmasının anahtar boyutu 64 bittir. 64 bitlik anahtar verisi PC-1 (Permuted Choice - 1) işleminden geçerek 64 bitlik veriyi 56 bit'e düşürür. PC-1 işlemi Şekil 2.19' da gösterilmiştir.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Şekil 2.19: PC-1 (Permuted Choice - 1)

Bu işlemin çıktısı olan 56 bitlik veri ikiye bölünerek C_0 ve D_0 oluşturmaktadır. İçerisinde 28 bit barındıran C_0 sola bir adım kaydırma işlemi yapar. D_0 ise sağa bir adım kaydırma işlemi yapar. Bu işlemden sonra sola bir adım kaydırılan C_1 ile sağa bir adım kaydırılan D_1 ile birleşdiyinde 56 bitlik bir veri oluşmuş oluyor ki bir sonraki işlem olan PC-2 işlemi 56 bitlik veriyi 48 bitlik veri haline çevirerek K_1 birinci turun anahtarını üretmiş oluyor. Şekil 2.20'de PC-2 (Permuted Choice - 2) gösterilmiltir.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Şekil 2.20: PC-2 (Permuted Choice - 2)

DES algoritmasının tur sayısı 16 olduğu için anahtar üretimi algoritmasının tur sayısı 16'dır. C_1 ve D_1 gizli anahtar olan K_1anahtarını ürettikten sonra C_1 bir adım sola kaydırılarak C_2'i, D_1 ise bir adım sağa kaydırıldıktan sonra D_2'i üretmektedir. Bir sonraki adımda C_2 ve D_2 PC-2 işlemine girerek 48 bitlik K_2 gizli anahtarını üretmektedir. Bu işlemler 16 kere tekrar edilir.

DES algoritması veriyi 64 bitlik bloklara parçalayarak şifreleme işlemini gerçekleştiriyor. Basitleştirilmiş olan SDES Simplified Data Encryptio System) algoritması ise şifrelenek olan veriyi 32 bitlik bloklara parçalayarak şifreleme işlemini gerçekleştirmektedir. DES algoritmasının bir üstü ise 3DES (Triple Data Encryption System) şifrelenecek veriyi 128 bitlik bloklar halinde şifreleme

işlemini yapıyor. 1990'lı yıllarının sonunda DES algoritmasına Kaba Kuvvet (Brute Force attack) saldırısı yapılarak kırılmıştır.

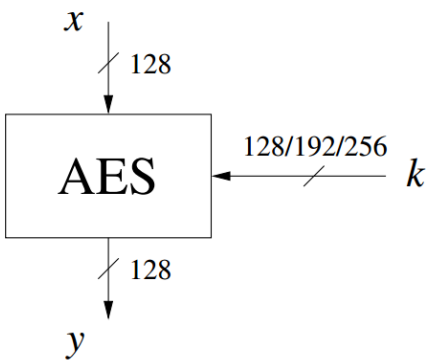
2.1.2.7 Gelişmiş Şifreleme Standardı (Advanced Encryption Standard)

1990'lı yılların sonlarında DES şifreleme algoritması Kaba Kuvvet saldırılarına karşı direnç gösterememesi sonucunda söz konusu olan bu algoritmayı geliştirilerek 3DES algoritması öne sunuldu. Bu algoritmanın DES algoritmasından farkı ise orijinal metni 128 bitlik bloklar halinde şifrelemeyi gerçekleştiriyor. 3DES algoritması DES algoritmasını 3 kere tekrarlayarak orijinal metni şifreler ve şifreleme işlemi için gizli anahtarın boyutu ise 192 bit olması gerekiyordu. Geliştirilmiş 3DES algoritması bir önceki versiyona göre daha karmaşık olması nedeni ile donanımsal olarak şifreleme işlemide DES algoritmasına göre 3 kat daha yavaş performans sağlıyordu. Bu tür sorunların giderilmesi amacıyla Ulusal Standartlar ve Teknoloji Enstitüsü (National Institude of Standards and Technology) yeni bir şifreleme sistemi geliştirilmesi talebinde bulundu. 1997 senesinde bu talep üzerine 15 farklı şifreleme sistemi incelenmeye alındı. Bu şifreleme sistemleri "CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC,LOKI 97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, Twofish" olarak kayda alınır. İleri sürülen bu 15 şifreleme algoritmasından 5 algoritma seçilerek olumlu ve olumsuz deyerlerine göre seçilerek Geliştirilmiş Şifreleme Standardı belirlenir.

1. Rijndael: 86 olumlu, 10 olumsuz
2. Serpent: 59 olumlu, 7 olumsuz
3. Twofish: 31 olumlu, 21 olumsuz
4. RC6: 23 olumlu, 37 olumsuz
5. MARS: 13 olumlu, 84 olumsuz

Böylece Rijndael algoritması AES algoritması adıylada tanınmaktadır. AES algoritması günümüz teknolojilerin bütün alanlarında kullanılmaktadır bunlara örnek olarak ABD hükumeti, ticari sistemler, WEB tarayıcılar ve başka bunun gibi teknolojiler güvenlik amacıyla AES şifreleme algoritmasını kulannmaktadır.

AES şifreleme sistemi bir Blok şifreleme sistemidir ki orijinal mesajı 128 bitlik bloklar halinde şifreler. Şekil 2.21'de AES şifreleme sisteminin genel yapısı gösterilmiştir.



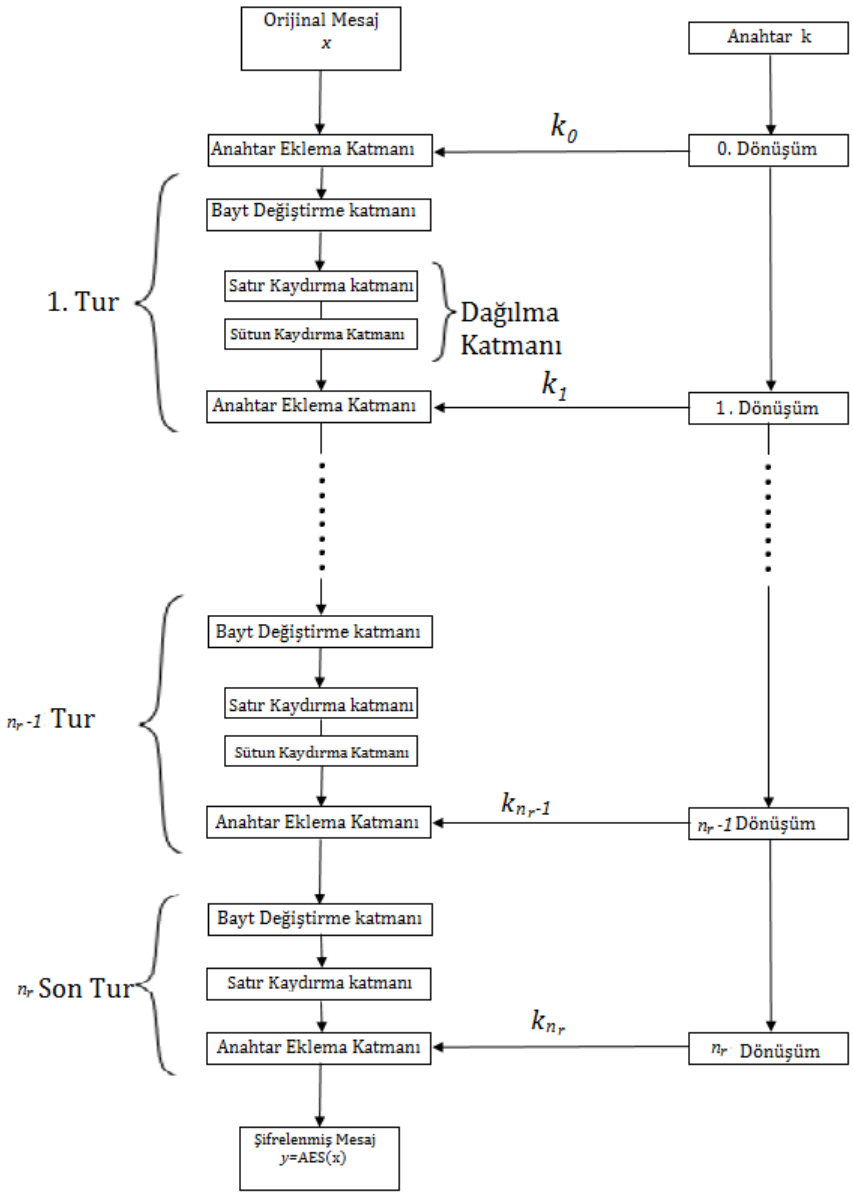
Şekil 2.21: AES şifreleme sistemi genel yapısı

AES şifreleme sistemi genel yapısından da görüldüğü gibi gizli anahtar olarak anahtarın boyutu 128 bit, 192 bit, 256 bit olabilmektedir. Gizli anahtarın boyutu yükseldikce şifreleme sisteminin turlarıda artmaktadır Çizelge 2.3'de anahtar boyutuna göre algoritmanın tur sayısı gösterilmiştir.

Çizelge 2.3: Anahtar boyutuna göre AES algoritmasının tur sayısı

Anahtar boyutu	Tur sayısı
128 Bit	10
192 Bit	12
256 Bit	14

AES şifreleme sisteminin her bir tur kendi içerisinde 3 farklı katman bulunmaktadır bu katmanlar Şekil 2.22'de gösterilmiştir



Şekil 2.22: AES genel diagramı

AES algoritmasının genel diagramındaki katmanlar anahtar boyutuna göre deyişen tur kadar tekrarlanmaktadır. Her turda gerçekleştirilen işlem katmanlarını daha detaylı göstericek olursak.

Anahtar Ekleme Katmanı- bu katmanda Anahtar Üretimi algoritmasının birinci turundan üretilmiş anahtar ile şifrelenecek orijinal mesaj XOR işlemine tabi tutulur.

Bayt Değiştirme Katmanı- bu katmanda anahtar ekleme katmanının çıktısı olan 128 bit yani 16 byte veriler onaltılık sisteme çevrilerek 4x4 matris şeklinde dizilir ve Şekil 2.23 de gösterilen S-kutuları ile ilgili onaltılık veri ile

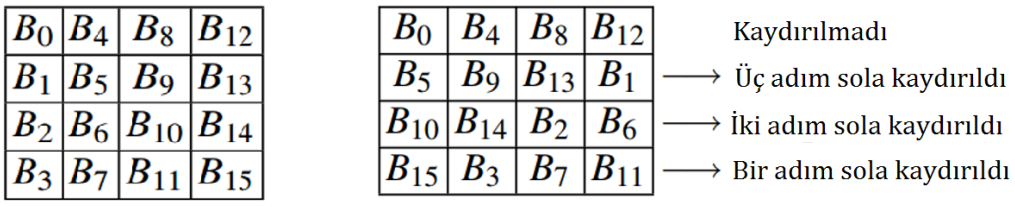
değiştirilir, örnek olarak $A_i = (C2)_{16}$ verisini S kutusundaki ilgili veri $S((C2)_{16}) = (25)_{16}$ olarak gösterilmiştir.

		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 2.23: S-kutusu (Substitution Box)

S-kutusu ile 4x4 matrisindeki bütün veriler deyiştirilir ve bir sonra ki katmana bu katmanın çıktısı gönderilir.

Satır Kaydırma Katmanı- bu katmanda bir önceki katmandan üretilmiş olan 4x4 boyutundaki matrisi çembersel olarak kaydırılır. Kaydırma işlemi ise 4x4 matrisinin ikinci satırından başlar. Ikinci satır üç adım sola, üçüncü satır iki adım sola, dördüncü satır ise bir adım sola kaydırılır. Bu işlem Şekil 2.24' de gösterilmiştir.



Şekil 2.24: Satır Kaydırma işlemi

Kolon Karıştırma Katmanı- bu katman da ise Satır kaydırma katmanının çıktısı 4x4 boyutundaki matris ile GF (2^8) ile üretilmiş olan sabit bir matris ile çarpılır. Şekil 2.25' de bu işlem gösterilmiştir.

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

Şekil 2.25: Kolon karıştırma işlemi

Anahtar Ekleme Katmanı- bu katmanda Kolon karıştırma katmanından elde edilen karıştırılmış 4x4 matrisi ile anahtar üretimi algoritmasının birinci dönüşüm anahtarı matris şeklinde dizilerek XOR işlemine tabi tutulur.

AES şifreleme algoritmasının anahtar boyutu 128 bit olursa eğer bu katmanlardaki işlemler 10 kere tekrar eder, fakat sonuncu turun diğer turlardan farkı *Sütun Karıştırma katmanı* devreden çıkartılır. Bir sonraki eylem bir sonraki 128 bit bloğu aynı katmanlardan geçerek şifrelemektir.

Gelişmiş Şifreleme Standardı Anahtar Üretimi

AES algoritmasının şifreleme işlemi başlamadan önce anahtar üretimi algoritması başlatılarak AES algoritmasının her turu için gereken anahtar parçacığını üretilir. Anahtar üretimi için girdi olarak kullanılan 128 bitlik anahtarın byte karşılığı 16 byte'dır ki bu girdiler onaltılık sisteme çevrilerek 4x4 boyutunda bir matris şeklinde dizilir (Çizelge 2.4).

Çizelge 2.4: Girilen Anahtarın onaltılık sisteme çevrilerek 4x4 matrisi

2b	28	Ab	09
7e	Ae	F7	Cf
15	D2	15	4f
16	A6	88	3C

Girilen anahtar 4x4 boyutundaki matris haline getirildikten sonra son kolonun ilk elemanı, son elemanının yerine geçer.

Çizelge 2.5: Girilen Anahtarın onaltılık sisteme çevrilerek 4x4 matrisin eleman değişimi.

2b	28	Ab	CF
7e	Ae	F7	4f
15	D2	15	3c
16	A6	88	09

Çizelge 2.5'den elde edilen matrisin son kolonu ele alınır ve belirlenmiş S kutusundaki (Şekil 2.26) elemanlar ile yer deyiştirilir.

									y							
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	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
x	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 2.26: S-Kutusu

Şekil 2.26'da gösterilen S-kutusundaki değerler ile son kolonu değiştirilmiş 4x4 matrisim son kolonu değiştirilir. Çizelge 2.6'da değiştirilmiş matris gösterilmiştir.

Çizelge 2.6: S-kutusu ile yer değiştirmiş 4x4 matrisi

2b	28	Ab	8a
7e	Ae	F7	84
15	D2	15	EB
16	A6	88	01

S-kutusu ile yer değişme işleminde çıkan 4x4 matrisinin ilk kolonu, son kolonu ve belirlenmiş 10x4 matrisinin (Şekil 2.27) ilk kolonu XOR işlemine tabi tutulur.

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

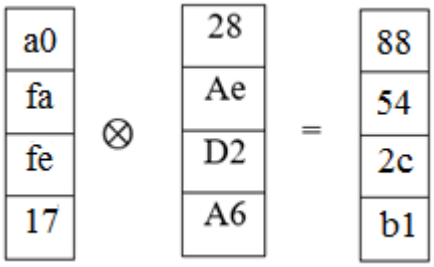
Şekil 2.27: RCON matrisi

RCON sabit matrisinin ilk kolonu, değiştirilmiş 4x4 matrisinin ilk kolonu ve son kolonu XOR işleminden geçirilerek Şekil 2.28'deki sonuç elde edilir

2b		8a		01		a0
7e	⊗	84	⊗	00	=	fa
15		eb		00		fe
16		01		00		17

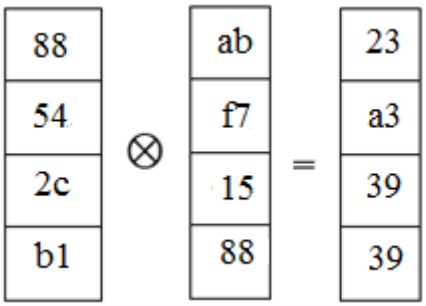
Şekil 2.28: RCON ilk kolon, 4x4 matris ilk kolon ve son kolon XOR işlemi

Şekil 2.28'de gösterilmiş olan XOR işleminin sonucundan elde edilen sonuç AES algoritmasının birinci turu için üretilecek olan 4x4 matrisinin birinci kolonudur. İkinci kolonun üretilmesi için ise Şekil 2.28'de gösterilen XOR işleminin sonucu ile ilk 4x4 matrisinin ikinci kolonu XOR işlemine tabi tutulur bu işlem Şekil 2.29 da gösterilmiştir.



Şekil 2.29: birinci turun 4x4 matrisinin birinci kolonu ile ilk 4x4 matrisinin ikinci kolonunun XOR işlem sonucu

Şekil 2.29'de gösterilen işlemin sonucu birinci tur için üretilecek olan 4x4 matrisinin ikinci kolonudur. Birinci tur için üretilecek matrisin üçüncü kolonunun üretilmesi için Şekil 2.29 işleminin sonucu ile ilk matrisin üçüncü kolonu XOR işlemine tabi tutulur.



Şekil 2.30: birinci turun 4x4 matrisinin üçüncü kolonu ile ilk 4x4 matrisinin dördüncü kolonunun XOR işlem sonucu

Şekil 2.30 gösterilen işlem sonucu ile birinci tur için üretilecek 4x4 matrisi Çizelge 2.7 de gösterilmiştir

Çizelge 2.7: AES algoritmasının birinci tur için üretilen anahtarı.

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

AES algoritmasının birinci tur için üretilen anahtarından sonra ikinci tur için anahtar üretimi başlatılacaktır. Tur anahtarı yukarıdaki şekilde 10 defa tekrarlanarak her tur için üretilir.

2.2 Açık Anahtarlı Şifreleme

Kriptografinin bir başka bölümü olan şifreleme tarzı olan Açık anahtarlı şifrelemenin bir başka adı Asimetrik şifrelemedir. Simetrik şifreleme (Gizli anahtarlı şifreleme) sistemleri ile Asimetrik şifreleme sistemdeki temel fark haberleşme esnasındaki şifrelemeyi hem daha güvenli hem daha rahat hale getirir. Bu şifreleme sistemini Şekil 2.31'de gösterilen Aslı ve Burağın güvensiz kanal üzerinden haberleşmesinden anlatıcak olursak.

Şekil 2.31: Açık anahtarlı şifreleme.

Açık anahtarlı şifreleme sisteminin esas kuralı her iki tarafın iki anahtarı olmak zorundadır dolayısıyla Şekil 2.1'de gösterildiyi gibi Aslı orijinal metni Burağın her kes tarafından bilinen açık anahtarıyla P_{Burak} şifreleyerek $c = E\ (P_{Burak}, m)$ şifrelenmiş mesajı Burağa gönderir. Burak bu şifrelenmiş mesajı c'i elde ettikden sonra deşifreleme işlemini $m=D(S_{Burak}, c)$ kimsenin bilmediği kendisine ait gizli anahtar olan S_{Burak} ile yaparak orijinal mesajı elde etmiş olur. Şifreleme ve Deşifreleme işleminin bu şekilde olmasına güvenlik açısından bakılırsa "Ortadaki Adam" saldırısına karşı daha güvenlidir, dolayısıyla güvensiz olan kanalı dinleyen Ege burağın açık anahtarını elde edicek olsa bile şifrelenmiş mesajı deşifreleme işlemi başarısız olacaktır. Bu tür şifreleme ve deşifreleme işlemine rahatlık açısından bakıcak olursak, günümüz dijital haberleşme sistemleri milyonlarca kişinin bir birileri ile karşılıklı haberleşmesini sağlamaktadır. Her kullanıcı haberleşmek istediği her bir kişi ile ortak gizli anahtarı aklında tutmak zorundadır, fakat Asimetrik şifreleme bu tür rahatsızlığı aradan kaldırmaktadır.

Asimetrik Şifreleme yöntemlerine örnek olarak aşağıdaki algoritmalar gösterilebilinir.

- RSA
- Diffie Hellman
- Merkle Hellman

Yukarıda gösterilen örnekler açık anahtarlı şifreleme yöntemleridir. Bu yöntemin Simetrik şifreleme yöntemine nazaran daha yavaş çalışmasıdır buna neden olarak şifreleme işlemi olarak önce sistemin hazırlanması, sonrasın şifreleme işleminin yapılması gerekiyor.

2.2.1 RSA Algoritması

RSA şifreleme algoritması Açık anahtarlı şifreleme yöntemidir 1977 senesinde Ron Rives, Adi Shamir, Lenard Aldiman tarafından geliştirilmiştir algoritmanın ismi bu geliştiricielerin soyisimlerinin baş harflerinden gelmektedir.

RSA şifreleme algoritması 3 basamaktan oluşmaktadır, Burak aşağıdaki işlemleri yaparak kendi Açık anahtarını ve gizli anahtarını belirleyecektir, sonrasında Açık anahtarını Aslıya göndererek gönderilecek mesajı bu anahtar ile şifrelemesi gerekiyor.

1. Sistemin hazırlanması başka bir deyişle gizli anahtar ve açık anahtar üretimi
    - İki tane asal sayı seçilmeli ve bu iki asal sayı p ve q olarak işaretlenmeli $p=3$ ve $q=11$
    - Seçilen iki asal sayıyı bir birine çarparak n değerini formülü ile $n=pq \Leftrightarrow 3 \times 11 = 33$ hesaplayarak alıcı taraf kendi açık anahtarlarından birisini belirlemiş oluyor
    - EKOK(p-1,q-1) <=> (2,10)=10
    - En Küçük Ortak Kat hesaplandıktan sonra öyle bir e değeri seçilmeli ki bu değer $1<e<10$ şartını sağlasın ve aynı zamanda bu 10 ile e kendi aralarında asal olmalı başka bir deyişle EBOB$(10,e)=1$, böylece seçilen e değerini 7 olarak belirlenir ki bu da alıcının bir diğer açık anahtarıdır
    - Alıcı tarafın özel anahtarının belirlenmesi için $de=1 \bmod 10$ formülü ile d değerini Ters Öklid algoritmasını kullanacağız.

Yapılan işlemler sonucunda çıkan denklem $7d=1 \bmod 10$ işlemini Ters Öklid algoritmasına göre hesaplamak gerekirse $10=7 \cdot 1+3$, sonra 3 değerini beraberliğin sol tarafına geçirecek olursak $3=10-7 \cdot 1$, aynı şekilde $7=3 \cdot 2+1$ bu işlemdeki 1 değerini beraberliğin sol tarafına geçirecek olursak $1=7-3 \cdot 2$ haline gelecektir bu işlemde 3 değerinin yerine $10-7 \cdot 1$ yaza biliriz $1=7-(10-$

$7 \cdot 1) \cdot 2$, bu işlemi sadeleştirecek olursak $1=7\text{-}10+7 \cdot 1 \cdot 2$. Beraberliğin sol tarafındaki değerleri d yerine koyucak olursak $7 \cdot 10=1 \bmod 20$ beraberliği doğru olmayacaktır, fakat beraberliğin sol tarafındaki 7 değerinin sayısı toplam 3 tane olduğu için $7 \cdot 3= 1 \bmod 20$ beraberliği doğru olucaktır. Ters Öklid algoritmasının sonucu olarak d değerini bulunur ki bu da alıcı tarafın gizli anahtarıdır, bir sonraki eylem RSA algoritmasının ikinci adımıdır.

2. Şifreleme işlemi.

Şifreleme işlemini formül olarak gösterilmesi gerekirse $C = m^e mod\, n$ şeklinde yapılır. Farz edelim ki Aslı Burağa "O" karakterini şifreleyerek göndermek istiyor. "O" harfinin alfabedeki sıra numarası ise "18" olarak belirler. Şifreleme işlemi formülüne göre $18^7 (mod\ 33) = 6$ olduğuna göre alfabedeki 6. Sıradaki harf "E" olduğuna göre bu şifrelenmiş mesajı Burağa gönderir.

3. Deşifreleme işlemi

Aslı tarafından Burağın açık anahtarı ile şifrelenen metnin Burağın gizli anahtarı ile deşifreleme işlemini yapmak için kullanılan formül $m = c^d mod\, n$ şeklindedir. Gelen şifreli mesajın "E" harfinin alfabedeki sıra numarasını c değerinin yerine koyduktan sonra deşifreleme işlemi gerçekleşmiş olucaktır. $m = 6^3 (mod\ 33) = 18$ böylece Aslının gönderdiği Şifreli mesaj deşifrelenmiş oldu.

2.2.2 Diffie Hellman Anahtar Değişimi

Asimetrik şifreleme yönteminde gönderici ve alıcı taraf aynı gizli anahtarı kullanarak şifreleme ve deşifreleme işlemini yapa bilirler. Simetrik şifreleme yönteminde ortak gizli anahtarın belirlenerek güvenli olan kanal üzerinden paylaşılarak şifreleme ve deşifreleme işlemi yapılır. Diffie Hellman Anahtar Değişimi işlemi ortak gizli anahtarın belirlenerek güvensiz kanal üzerinden güvenli şekilde karşı tarafa iletme tekniğidir. Bu teknik ilk defa 1976 yılında Whitfield Diffie ve Martin Helman tarafından geliştirilmiştir.

Diffie Helman tekniğini uygulamak için Aslı ve Burak kendi aralarında güvensiz kanal üzerinden asal sayı seçmesi gerekiyor ki bunlar $g=5$ ve $q=11$ olduğunu farzedelim. Sonra Aslı kendi özel numarasını rastgele olarak 14 seçer ve $5^{14} \bmod 11 = 9$ ve elde ettiği bu sonucu güvensiz kanal üzerinden Burağa gönderir. Sonra Burak kendi özel numarasını rastgele olarak 18 ve $5^{18} \bmod 11$

$= 4$ işlemin sonucunu Aslıya gönderir. Aslı bu sonucu elde ederek $4^{14}(mod\ 11) = 3$. Burak ise Aslıdan gelen *9* numarasını elde ederek $9^{18}(mod\ 17) = 3$ işlemini yapar. Böylece sürekli dinlenilen güvensiz kanal üzerinden ortak gizli anahtar belirlenmiş oldu.

2.2.3 Merkle Hellman Algoritması

Merkle Hellman açık anahtarlı şifreleme yöntemidir 1978 senesinde Martin Hellman ve Ralph Merkle tarafından geliştirilmiştir. Piyasaya sürüldüğü zamanlarda çok zekice tasarlandığı düşünülsede bu algoritma kırılmıştır. Merkle Hellman algoritmasının birinci adımı anahtar üretimi işleminde ilk olarak süperartan bir dizi belirlenmeli. $W=(w_1, w_2, \dots w_n)$ *n*- sıfırdan farklı doğal bir sayı olmalı örnek olarak $W= \{1, 2, 4, 8, 16, 32, 63, 127\}$,diziyi ele alalım. Bir sonraki adım olarak ise $\sum_{i=1}^{n} w_i$ hesaplanmalı. Ele aldığımız dizinin elemanlarının toplamı ise $\sum W = 253$, sonra öyle bir q değeri seçilmeli ki bu değer $q > \sum_{i=1}^{n} w_i$ şartını sağlamalı ve aynı zamanda $\sum_{i=1}^{n} w_i$ ile q kendi aralarında asal olmalı, burada $q=257$ olarak seçersek her iki şart sağlanmış olur, sonra öyle bir r değeri seçilmeli ki bu değer $1 < r < q$ şartını sağlamalı ve aynı zamanda r ile q kendi aralarında asal olmalı bu örneğimizde $r= 17$ değerini alıcak olursak her iki şart sağlanmış olur. Elde ettiğimiz $\sum W$, q, r değerleri bizim gizli anahtarımız olucaktır, bir sonraki aşama ise açık anahtarımızı hesaplamak olucaktır ki formül olarak $B_i = W_i * r\ (mod\ q)$ şeklinde devam edersek açık anahtarı elde etmiş oluruz verilen formüle göre örnek süperartan diziden açık anahtarı hesaplayalım.

$1 \cdot 17 \bmod 257 = 17$

$2 \cdot 17 \bmod 257 = 34$

$4 \cdot 17 \bmod 257 = 68$

$8 \cdot 17 \bmod 257 = 136$

$16 \cdot 17 \bmod 257 = 15$

$32 \cdot 17 \bmod 257 = 30$

$63 \cdot 17 \bmod 257 = 43$

$127 \cdot 17 \bmod 257 = 103$

$$127 \bullet 17 \bmod 257 \quad = 103$$

Böylece Açık anahtarımız $B = \{17, 34, 68, 136, 15, 30, 43, 103\}$ şeklinde olacaktır. Bir sonraki aşama ise şifreleme işlemini yapmak. Şifreleme işlemi yapmak için verilmiş orijinal mesaj ikili sistem haline getirilir $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ şifreleme işlemi formül olarak gösterecek olursak $C = \sum_{i=1}^{n} \alpha_i B_i$ şeklinde olucaktır. Verilmiş örnek üzerinden "R" orijinal mesajının şifreleme işlemi yapmak için orijinal mesajın ikili sistem karşılığı hesaplanmalı ki $"R" \Leftrightarrow bin = 01010010$ şeklinde olucaktır. Şifreleme aşamasında ise verilen formüle uygun olarak işlem aşağıdaki gibi olucaktır.

$$C = 17 \bullet 0 + 34 \bullet 1 + 68 \bullet 0 + 136 \bullet 1 + 15 \bullet 0 + 30 \bullet 0 + 43 \bullet 1 + 103 \bullet 0 = 34 + 136 + 43 = 213$$

Şifreleme sonucuna göre karşı tarafa gönderilecek şifreli mesaj "213" olucaktır. Alıcı taraf şifreli mesajı aldıktan sonra deşifreleme işlemi için kullanılıcak olan formül $C \cdot r^{-1} mod\ q$ şeklinde yazılır. Verilen örnekten yola çıkılacak olunursa $213 \cdot 17^{-1} mod\ 257$ şeklinde hesaplanmalı. Verilen formüle göre çarpma işlemini çarpmadan önce $17^{-1} mod\ 257$ modüler tersi Ters Öklit algoritmasıyla hesaplanmalı. Bu algoritma aşağıdaki gibidir.

$$x = 17^{-1} mod\ 257 \Leftrightarrow x = \frac{1}{17} mod\ 257 \Leftrightarrow 17x = 1\ mod\ 257.$$

$$257 = 17 \bullet 15 + 2 \Leftrightarrow 2 = 257 - 17 \bullet 15$$

$$17 = 2 \bullet 8 + 1 \Leftrightarrow 1 = 17 - 8 \bullet 2 \Leftrightarrow 17 - (257 - 17 \bullet 15) \bullet 8 = 17 - 257 \bullet 8 + 17 \bullet 15 \bullet 8 \Leftrightarrow 121 \bullet 17 - 257 \bullet 8$$

$$257 \bullet 8\ (mod\ 257) = 0$$

$$17 \bullet 121 (mod\ 257) = 1 \Leftrightarrow 121 = 17^{-1} mod\ 257$$

Ters Öklit algoritmasına göre $17^{-1} mod\ 257$ bu işlemin tersi $121\ mod\ 257$ olduğunu elde ettik. Deşifreleme formülüne göre 213*121 (mod 257)=73 sonucunu elde ettik. Şifrelenmiş metnin orijinal halini ikili sistem şeklini bulmak için W dizisindeki $73 > W_i$ şartını sağlayan bütün rakamları seçip aşağıdaki işleme tabi tutuyoruz.

73-63=10

10-8=2

2-2=0

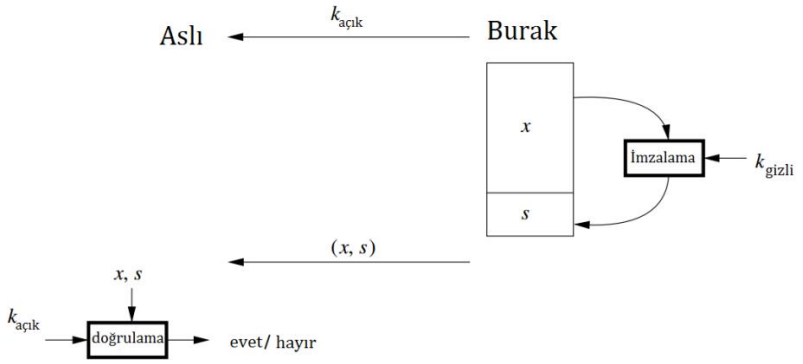
Yukarıda yapılan işlemde kullanılan W kümesinin karakterlerinin yerine "1" kullanılmayan karakterin yerine ise "0" koyarsak deşifreleme işlemi gerçekleşmiş olucak ki Çizelge 2.8 gösterilmiştir

Çizelge 2.8: Merkle Hellman Deşifrelenmiş Metin

W_i	1	2	4	8	16	32	63	127
R	0	1	0	1	0	0	1	0

2.2.4 Dijital İmzalama Algoritması

Açık anahtarlı şifreleme işleminin temel çalışma prensibi güvensiz kanal üzerinden iletişimde olan taraflarının her birinin hem her kes tarafından bilinen anahtarı ki bu anahtar sadece şifreleme işlemini gerçekleştiren anahtardır, hem de kimsenin bilmediği gizli anahtar ki bu anahtar şifrelenmiş mesajı deşifrelenmesi için kullanılır. Açık anahtarlı şifreleme işlemi ortadaki adam saldırılarına karşın güvenilir iletişim sağlamaktadır. Açık anahtarlı şifreleme işleminde her kes tarafından belirlenen anahtarın kopyalanmaması için gereken öğe Dijital imzalama işlemidir, dolayısıyla gönderilen şifrelenmiş mesajda gönderen tarafın kendine ait dijital imzasının olması gelen mesajın doğru adresden geldiği doğrulanmış olur. Şekil 3.5 de Dijital imzalama işleminin diagramı gösterilmiştir.



Şekil 2.32: Dijital imzalama işleminin diagramı

Dijital imzalama işleminden görüldüğü gibi Burak Aslı'ya x mesajını gönderirken Burağın gizli anahtarı olan k_{gizli} ile x mesajı imzalama fonksiyonuna girer ve (x,s) çıktısını aslıya gönderir. Şifrelenmiş mesajı alan Aslı mesajın imzasını doğrulamak için şifrelenmiş olan imzayı, şifrelenmiş mesajı, ve $k_{\text{açık}}$ anahtarı girdi olarak kullanmalıdır.

Dijital imzalama algoritmalarından en ünlü olanları aşağıdakilerdir.

- RSA Dijital İmzalama

Dijital imzalama tekniğinin sağladığı bir diğer avantajı ise inkar edememedir. Her iki taraf gönderdiği mesaja dijital imza koyduğu için yazılan mesajı kanıt olarakda kullanabilir olmasıdır.

2.2.4.1 RSA Dijital İmzalama Algoritması

RSA açık anahtarlı şifreleme algoritmasının dijital imzalama bölümü iletilen şifreli mesajların sertifikalanması, dolayısıyla gönderen tarafın şifreli mesaj üzerinde tarafa özel dijital imza yerleştirmek için geliştirilen teknikdir. Bu algoritmanın dijital imzalama bölümü ile birlikde 4 bölümü vardır. Bunlar aşağıdakilerdir.

Anahtar üretimi için

- iki tane p ve q asal sayıları seçilmeli $p=5$ ve $q=17$
- Mesajı alan taraf kendi açık anahtarını belirlemek için seçilen p ve q değerlerini çarparak açık anahtar çiftinden birincisini n elde etmiş oluyor $n=p \times q=5x17=85$
- EKOK(p-1,q-1)<=> (4,16)=16
- En Küçük Ortak Kat hesaplandıktan sonra öyle bir e değeri seçilmeli ki bu değer $1<e<16$ şartını sağlasın ve aynı zamanda bu 16 ile e kendi aralarında asal olmalı başka bir deyişle $EBOB(16,e)=1$, böylece seçilen e değerini 11 olarak belirlenir ki bu da açık anahtar çiftinin bir diğeridir
- Alıcı taraf kendi gizli anahtarını belirlemesi için gereken formül $de=1 \bmod 16$ şeklindedir bu formüle göre $11 \times d= 1 \bmod 16$ Ters Öklit algoritması ile hesaplayarak d değerinin 3 olduğunu görüyoruz.

Şifreleme işlemi

Şifreleme işlemi için gereken formül $C=m^e \ mod \ n$ şeklindedir. Mesaj gönderen taraf Mesajı şifrelemek için Örneğin "18" rakamını alıcı tarafın yayınladığı açık anahtaar çiftlerini *e,n* formülde uygun olarak yerlerşne koyulursa $C=18^{11} \ mod \ 85$ şeklinde hesaplayarak şifreli mesajın şifreli mesajın 52 rakamını elde etmiş oluyoruz.

İmzalama İşlemi

Gönderen taraf şifreli mesajın olan "52" harfini göndermeden önce mesajın yanında birde dijital imzalama işlemi yaplmalıdır. Dijital imzalama işlemi $S=M^d \ mod \ n$ formülünü kullanılıcak. Buradaki *M* orijinal mesaj, *e, n* alıcı tarafın açık anahtar çifti. Verilen örneğe göre gönderilecek mesaja dijital imza atmak için $S= 18^3 \ mod \ 85$ olarak hesaplarsak dijital imza *52* olacaktır. Böylece gönderen taraf şifreli mesaj ile beraber dijital imzayıda gönderecektir.

Deşifreleme İşlemi

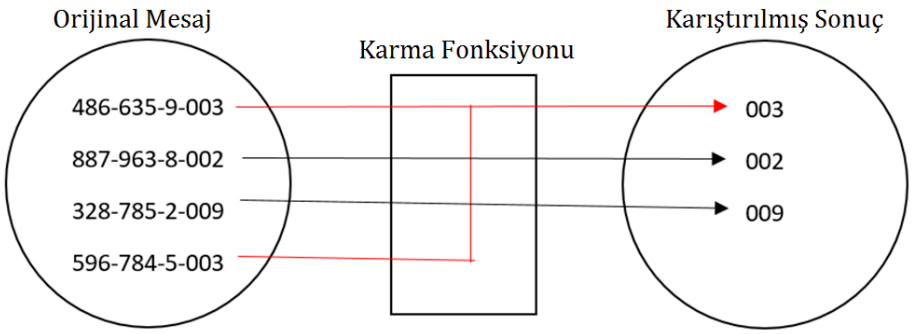
Şifrelenmiş olan mesajı deşifrelemek için kullanılan fomrül $M=c^d mod \ n$ şeklindedir. Buradaki *c* şifreli mesaj, *d* ise alıcı tarafın gizli anahtarıdır, *n* ise alıcı tarafın açık anahtarıdır. Verilen örnekten yola çıkılacak olunursa $M=52^3 mod \ 85$ hesapladığımızda sonuç 18 olarak çıkacaktır ki bu da orijinal metnimiz olacaktır.

İmza Onaylama

Alıcı taraf gönderilen mesajın doğruluğunu kontröl etmek için mesajın üstünde gelen şifreli dijital imzayı da deşifrelenmesi gerekiyor. Bu işlem için gereken formül ise $D= S^e mod \ n$ şeklindedir. Bu işlemde *S* dijital imza, *n* alıcının açık anahtarı, *e* alıcının açık anahtarıdır. Bu işlemin sonucu, eğer orijinal mesajı verirse mesajın dijital imzasının doğruluğu isbat edilmiş olur. Verilen örnekten yola çıkılırsa $D=52^7 mod \ 85$ işleminin sonucu 18 olacaktır ki bu rakam ile deşifrelenmiş mesaj numarası aynıdır.

2.3 Karma Fonksiyonları (Hash Functions)

Karma fonksiyonları Kriptografinin önemli parçasından biri olup orijinal mesajı tek yönlü şifrelemesini yapmak için kullanılan fonksiyonlardır. Karma fonksiyonlarının temel çalışma prensibi farklı uzunluklu veriler üzerinde belli işlemler yaparak belirlenmiş sabit uzunluklu veri haline getirmektir. Bu fonksiyonların temel amacı verilerin bütünlüğünün kontrolüdür. Bir başka avantajı karma fonksiyonlarının çıktısı olan veriler veri tabanında arama işleminin hızlı şekilde yapılmasıdır, bu avantajı sağlayan sebebi ise bu veriler şifrelendikten sonra benzerlik oranları çok düşük olduğu için karşılaştırma işleminin hızlı çalışmasıdır. Bu tür fonksiyonların en çok kullanıldığı yerler sosyal ağlardaki kullanıcı girişi işleminin yapıldığı yerlerdir. Buna sebeb olarak ise sunuculardan her hangi birine saldırıda bulunarak veriler ele geçirilse bile verilerin çözümlenmemesi gösterilmektedir. Karma fonksiyonlarının dezavantajları belirli durumlarda karıştırılmış verilerin benzerliğinin çakışmasıdır Şekil 2.33 de çakışmanın örneği gösterilmiştir.



Şekil 2.33: Karma fonksiyonlarının Çakışması.

Şekil 2.33'de karma fonksiyonu girdilerin sağdan 3 basamağını kullanarak karıştırma eyleminde birinci girdi ile sonuncu girdinin farklı olmasına rağmen çıktının aynı olması karma fonksiyonunun çakışmasıdır ki, bu gibi durumları minimuma indirgenmiş fonksiyonlar iyi performanslı fonksiyon olarak algılanılır.

Modern teknolojilerde en sık kullanılan karma fonksiyonlar aşağıdakilerdir.

- MD-5 (Mesaj Özeti - Message Digest)
- SHA 1 (Güvenli Karma Algoritması-Secure Hashing Algorithm)

Modern teknolojiler iki algoritmayıda kullanmaktadır.

2.3.1 MD-5 Algoritması (Message-Digest algorithm 5)

MD-5 Karma fonksiyonu MIT (Massachussetts Institute of Technology) kriptografcı Ronald Rivest tarafından 1991 senesinde, MD-4 karma fonksiyonunun geliştirilmiş versiyonudur ve MD-4 e göre daha karmaşık ve daha düşük performans sergilemektedir. Bu karma fonksiyonları en çok veri bütünlüğünün kntrolü için kullanılmaktadır. Girdi olarak şifrelenmek istenen verinin boyutu ne kadar olursa olsun MD-5 algoritması 128 bitlik karmaşık karakter üretecektir. Girilen veride küçük değişiklikler yapılırsa tekrar MD-5 algoritması tamamen farklı bir 128 bitlik karmaşık bir veri üretecektir. Bu algoritmanın Çakışma olasılılığı 2^{128} kadardır. Bu şifreleme algoritması 64 turdan ve her turda 3 temel adımdan ibarettir.

Birinci adım *Dolgulama* adımı. Bu adımda eğer şifrelenecek olan verinin boyutu yetersiz olucaksa MD-5 algoritmasının dörd temel bloğunu dolduracak kadar dolgulama işlemi yapılır. Bu işlem ise veriler ikili sisteme çevrilerek sonuna "1" eklenir ve sonrasında bloklar dolana kadar "0" eklenir.

İkinci adım 512 bitlik bloklar kapasitesi 128 bit olan 4 küme içerisine yerleştirilir bu kümelerdeki verileri onaltılık sisteme çevirerek göstericek olursak

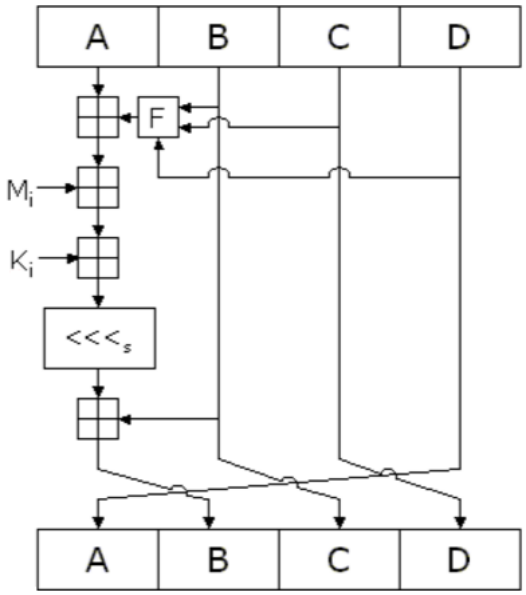
A= 01 23 45 67

B= 89 ab cd ef

C= fe dc ba 98

D= 76 54 32 10

olarak dizilir. Şekil 2.34 de MD-5 şifreleme algoritmasının diagramı gösterilmektedir



Şekil 2.34: MD-5 algoritmasının diagramı

Üçüncü adım olarak ise diagramdan da görüldüğü gibi *B, C,* ve *D* kümeleri *F* fonksiyonuna girer. *F* fonksiyonunda ise bu kümedeki veriler kendi aralarında "⊗" XOR, "∧" AND, "∨" OR, "¬" NOT işlemlerine tabi tutulurlar

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \neg D)$$

"⊗" XOR, "∧" AND, "∨" OR, "¬" NOT işlemlerinin çalışma prensibi Şekil 2.35 de Çizelgelar halinde gösterilmiştir.

XOR işlemi		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

AND işlemi		
A	B	A^B
0	0	0
0	1	0
1	0	0
1	1	1

OR işlemi		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT işlemi	
A	$\overline{A}$
0	1
1	0

Şekil 2.35: F fonksiyonu içerisindeki işlemler.

B kümesi F fonksiyonuna girerken aynı zamanda bir sonraki tur için C pozisyonunu almaktadır. C kümesi F fonksiyonuna girerken aynı zamanda bir sonraki tur için D pozşsyonunu almaktadır. D kümesi F fonksiyonuna girerken aynı zamanda bir sonraki tur için pozisyonunu almaktadır.

Üçüncü adım ise F fonksiyonundan çıkan bit ile *A* kümesinden gelen bitler toplanır ve bu sonucun mod 2^{32} bulunur. Formül olarak yazıcak olursak

$N_{A_{bit}} + N_{F_{çıktı}} = R \; mod \; 2^{32}$ şeklinde olur. Bu işlem sonucunda elde edilen bit ile orijinal mesajın bitleri M_i toplanır ve sonuç $mod\, 2^{32}$ göre hesaplanarak elde edilen bit ile ilgili tur için üretilmiş olan anahtarın bitleri toplanarak $mod\, 2^{32}$ göre hesaplanır. Buradan çıkan bitler sola doğru cari tur sayısı kadar kaydırılır ki burdan elde edilen bit ile *B* kümesinden gelen bitler toplanarak $mod\ 2^{32}$ göre hesaplanır. Buradan çıkan sonuç bir sonraki tur için *B* pozisyonunu alıcaktır.

Gösterilen 3 temel adım 64 defa tekrarlanarak 128 bitlik bir karma veri üretmektedir.

2.3.2 SHA Algoritması (Secure Hash Algorithm)

Karma fonksiyonlardan en çok kullanılanlardan biriside SHA algoritmasıdır. Bu algoritma 1993 senesinde NIST (National Institute of Standards and Technology) ve NSA (National Security Agency) tarafından geliştirilmiştir. SHA algoritması 5 farklı tipleri vardır bu tipler Çizelge 2.9 de gösterilmiştir ki bu tipler arasındaki fark anahtar uzunluklarına göre farklı sabit boyutlu karmaşık veriler üretiyorlar.

Çizelge 2.9: SHA algoritması tipleri arasındaki farklar

Algoritma	Anahtar Boyutu	Blok Boyutu [bit]	Çıktı [bit]	Tur Sayısı	Çakışma
SHA-1	128	512	160	80	No
SHA-2	224	512	224	64	No
SHA-2	256	512	256	64	No
SHA-2	384	1024	384	80	No
SHA-2	512	1024	512	80	No

SHA-1 algoritması yapı olarak MD-4 ve MD- 5 yöntemlerine daha çok benzemektedir ki 128 bitlik anahtar boyutu ile istenilen boyutlu girdiyi 160

bitlik karmaşık veri üretecektir. MD-5 ile SHA- algoritması arasındaki fark ise çakışma olayına rastlanmamasıdır.

SHA- 1 Algoritması 80 turdan ibaret olub her turda 5 temel adım gerçekleşmektedir. Birinci adım *Dolgulama*- işlemi yapmaktadır. Bu işlem MD-5 de olduğu gibi eğer girilen veri SHA-1 algoritmasının 5 temel bloğunu tam doldurmuyor ise veri ikili sisteme çevrilerek en sonuna "1" atanır ve 5 tane 512 bitlik kapasitesi olan blokları doldurulana kadar "0" eklenir

Dolgulama işleminden sonra ikinci adım olarak olarak *Kümeleme* işlemi yapılır girilen veri beş ayrı bloklara ayrılır ve her bloğun boyut kapasitesi 512 bit'dir. Her kümedeki verileri onaltılık sistemde gösterecek olursak

A = 0x 67 45 23 01

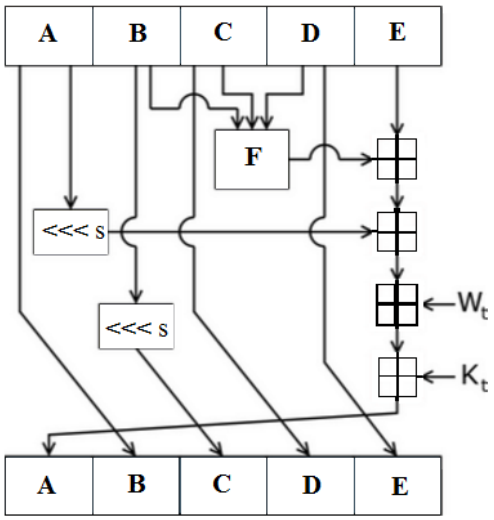
B = 7b ef cd ab 89

C = 56 98 ba dc fe

D = 4b 10 32 54 76

E = 9m c3 d2 e1 f0

Yukarıda gösterilen kümeleme işlemi tamamlandıktan sonra üçüncü adım olarak bu kümelerdeki veriler *F* fonksiyonuna girerler. Şekil 2.35 de SHA-1 algoritmasının diagramı gösterilmiştir.



Şekil 2.36: SHA-1 algoritmasının diagramı

SHA-1 algoritmasının diagramından da görüldüğü gibi *B, C ve D* bloklarındaki veriler *F* fonksiyonuna girer. F fonksiyonu içerisinde ise bu veriler kendi

aralarında "⊗" XOR, "∧" AND, "∨" OR, "¬" NOT işlemlerine tabi tutulurlar.

$$F_t(B, C, D) = (B \wedge C) \vee (\neg B \wedge D), \text{ for } t=0 \text{ to } 19$$

$$F_t(B, C, D) = B \otimes C \otimes D, \text{ for } t=20 \text{ to } 39$$

$$F_t(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D), \text{ for } t=40 \text{ to } 59$$

$$F_t(B, C, D) = B \otimes C \otimes D, \text{ for } t=60 \text{ to } 79$$

Dürdüncü adım olarak ise Şekil 2.35'den görüldüğü gibi *B* bloğundaki veriler hem *F* fonksiyonuna girer hemde ilgili tur numarası kadar sola kaydırılarak bir sonraki tur için *C* bloğuna girer. *C* bloğundaki veriler ise bir sonraki tur için *D* bloğuna girer. *D* bloğundaki veri ise bir sonraki tur için *E* bloğuna girer.

Beşinci adım olarak ise *E* bloğundaki veriler ve *F* fonksiyonundan çıkan veriler ikili sisteme çevrilerek toplanır ve *mod* 2^{32} göre hesaplanır. *A* bloğundaki veriler hem bir sonraki tur için *B* bloğuna girer hemde ilgili tur numarası kadar sola kaydırılarak *E* ve *F* modüler toplanması sonucunda çıkan veriler ile toplama işlemine girer bu işlemin sonucundan çıkan veriler *mod* 2^{32} göre hesaplanır. Bu işlemden çıkan veriler genişletilmiş olan orijinal metinin ikili sisteme çevrilerek toplanır ve bu sonuç 2^{32} göre hesaplanır. Burdan çıkan veriler ile ilgili tur için üretilen anahtar verileri ikili sisteme çevrilerek toplanır ve bu toplam sonucu 2^{32} göre hesaplanır. Bu hesaplamadan çıkan veriler ise bir sonraki tur için *A* bloğuna girer. Bu yapılan işlemler eğer anahtar uzunluğu 128 bit olursa 80 defa tekrarlanarak 512 bitlik karmaşık veri üretilecektir.

2.4 Steganografi

Güvenlik alanının bir diğer bölümü olan Steganografi iletişim esnasında Ortadaki Adam saldırılarına karşı önlem almak için geliştirilmiş teknikdir. Steganografi ile Kriptografi arasındaki fark ise Kriptografide gönderilecek orijinal mesaj belli bir matematiksel işlemlerden geçirilerek karıştırılır ve karşı tarafa gönderilir, karşı taraf şifrelenmiş olan bu mesajı çözümleyerek okunur hale getirir. Steganografideki amaç ise orijinal mesajı dönüştürerek önlem almak değil mesajın varlığını tamamen gizleyerek göndermektir. Orijinal Mesaj bir taşıyıcıya dikkat çekmeyecek şekilde gizlenerek gönderilir.

Steganografi Yunan kelimesidir ki "*Steganos*" gizlenmiş ve "*Graphien*" yazı kelimelerinin birleşiminden oluşmuştur. Türkçe karşılığı ise "Veri Gizleme" olarak düşünülebilir. İlk steganografiye Eski Yunan tarihçisi ve yazar Heredot, "Heredot Tarihi" eserinde rastlanmıştır. MÖ 484 – MÖ 425 senelerinde eski Yunan ve Pers imparatorluğu savaş halindedir. Pers imparatorluğu yöneticileri kendi aralarında güvenli iletişim yapmak için sadık kölelerinin saçlarını kazıtarak kafasına yazmıştır, saçları yeniden uzadığında köleyi karşı tarafa gönderme yöntemlerini kullanırlardı.

Steganografiye bir başka örnek olarak Amerikan Devrimi zamanlarında güvenli haberşeşmeyi sağlamak için görünmez mürekkeblerden faydalanmıştır. Görünmez mürekkeb olarak bir çok sıvı kullanılmıştır. Bunlara örnek olarak limon suyu, soğan suyu, süt, idrar, meyve suları ve sirke kullanılıyordu ki bu tür sıvılar ile yazılar yazılıp karşı tarafa gönderilir ve karşı taraf bu mektubu ısıtarak o sıvıların rengnini koyulaştırılırdı. Şekil 2.36'da örnek görünmez mürekkebi görünür hale getirilmesi gösterilmiştir

Şekil 2.37: Görünmez metinin görünür hale getirilmesi

İkinci dünya savaşı zamanlarında bir çok yeni teknikler kullanılmıştır bunlardan biriside fotorağfçılığın gelişimi ile gelen "Mikro Noktalama" tekniğidir. Bu teknikle yazılan mesaj fotoğraf teknikleri ile 200 kat ve daha da küçültülerek bir nokta boyutuna getirilerek dergi gazete ya da her hangi taşıyıcılar üzerine yerleştirilib gönderiliyordu. Şekil 2.37'de Mikro noktalamanın 1 milimetre boyutuna kadar küçültülmüş olan hali ve büyütülmüş hali gösterilmiştir.

Şekil 2.38: Mikro Noktalama

Teknoloji dünyasının gelişimi sonucunda steganografi de bu dünyaya giriş yapmış oldu. Özellikle internetin gelişimi yoğun bir bilgi akışını gerçekleştirdi. Normal hayatta verilerin saklanarak gönderilmesi internet dünyasında oldukça popüler olmaya başladı. İnternet dünyasında steganografinin kullanımı, sadece harflerin saklanması ile yeterli olmayıp bu işlemi Resimler üzerinde, ses kaydı üzerinde, video kayıtlarda da yapılmaktadır. Bu işlemler yapılarken steganografik algoritmalar 2 temel özelliğe dayanmalıdır.

1. Değişimin Farkedilmemesi- saklanılacak olan bilgi için öyle bir örtü verisi (cover data) seçilmeli ki yapılan değişikliler insan duyuları tarafından algılanabilicek konumda olmamalı. Bu özellik dikkate alınmaması durumunda örtü verinin birşeyler saklandığı sonucuna varılarak saklanmış mesaj çözümlenebilinir.
2. Saklanabilecek veri miktarı- örtü verisinde saklanıcak verinin boyutuna dikkat edilmesi gereken önemli noktadır ki eğer saklanılacak veri miktarı gereğinden fazla olursa örtü verisinin bozulması kaçınılmaz sonuç olucak, dolayısıyla örtü verisinin altındaki gizliliği insan duyuları farkedecektir.

Günümüz teknolojisinde geliştirilen Steganografi teknikleri yukarıda saydığımız özelliklere dayanmaktadır. Dijital ortamlarda yapılan teknikler inceleyecek olursak.

Metin Steganografi (Text Steganography)-Steganografi bu tekniği Teknolojinin henüz gelişmediği zamanlardan bu tarafa vardır. Dijital ortamlarda aynı şekilde kullanılmaktadır. Örneğin:

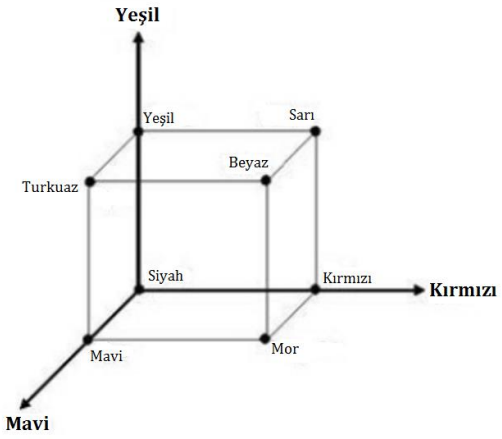
After **T**he **T**heater, **A**ll **C**lients **K**eep **A** **T**ab **D**own **A**t **W**esley's **N**ook

ATTACKATDAWN

(Mesaj kelimelerin ilk harfleri şeklinde kodlanmış)

*Açık Alan Steganografi (*Open Space*)*- Bu yöntemler kelimeler arasındaki boşluklar ile veriler saklanılarak karşı tarafa gönderilir. Örneğin kelimeler arasındaki boşluklar hesaplanarak normal metin içerisinde başka bir veri saklanıla biliniyor. Kelimeler arasındaki tek boşluk "0", çift boşluk ise "1" olarak hesaplanır ve veri karşı tarafa gönderilir. Bu yöntemin bir dezavantajı tek bir kelime için uzun metin yazma gereği, bir diğeri ise bazı haberleşme sistemleri fazla boşlukları otomotik olarak temizler.

Görüntü Steganografi (İmage Steganography)- Steganografinin bu yöntemini kullanılması isteniyorsa dijital resimlerin yapılarını iyi şekilde benimsenilmesi gerekiyor. Dijital resimlerin her biri piksellerden oluşuyor ki bu piksellerdeki her reng, temel rengler olan Kırmısı, Yeşil, Mavi renglerinin karışımından oluşmaktadır. Dolayısıyla temel renglerin herbirinin boyutu 1 byte olduğuna göre her pikselin boyutu 3 byte olmaktadır. Şekil 2.37'de temel renglerden oluşan diğer renglerin 3 boyutlu ortamdaki pozisyonları gösterilmiştir.



Şekil 2.39: Reng Uzayı

Reng uzayı Şeklinde renglerin her hangi birisi ile 3 bitlik oynama yapılırsa insan gözü bu renglerdeki deyişimi farketmeycektir, dolayısıyla her hangi bir resimde veri saklamak için her pikselde kırmızı, yeşil ve mavi reng tonlarının her birinin byte karşılığı en fazla 255 byte'dır. Piksellerdeki kırmızı, yeşil ve mavi reng tonlarında en az önemli bitlerin yerine gizlenmesi gereken verinin bit karşılığındaki veriler yerleştirilir. En az önemli bit ise eğer renglerin değerleri "0" ile bitiyorsa burdaki "0" değerinin yerine gizlenmek istenen verinin bit karşılığının "1" değerini koyarsak pikseldeki değişiklik farkedilmeden veri resim içerisinde saklanmış olacaktır. Günümüz fotoğraf makinelerinin piksel sayısı ortalama olarak 800x600 kadardır ki buda 480.000 piksel anlamına gelir ve herbirinde 3 bitlik boş alan olduğuna göre 1.440.000 bitlik boş bir alan bulunmaktadır.

Ses Steganography (Audio Steganography) - Ses bilgilerinin dağıtımı ve içerdiği bilginin özelliği açısından Resim bilgilerine benzemektedir. Ses bilgileri içerisinde veri saklamak insan işitme sistemi yüzünden oldukca uğraş gerektiren bir işlemdir. Resimler içerisinde veri saklama yöntemi ile ses dosyaları içerisinde veri saklamak için ses dosyalarında değişikliği fark ettirmeyen bitlerin yerine saklanılacak veri gömülürse veri saklamış olan ses gürültülü hale gelmiş olucaktır. Ses steganografisi ile veriyi gizletme işleminin dezavantajı ise iletme esnasında nereden geldiği belli olmayan gürültüler sayesinde saklanmış olan veri zarar göre bilir. Bu tür kayıpları aradan kaldırmak için ise gönderilecek dosyaya yankı eklenir ve bu yankı geciktirilir ve geciktirilen aralığa ise saklanılacak veriler gömülerek gönderilir. Bu şekilde veri iletilirse veri kaybı yaşanmadan karşı taraf gizli veriyi elde eder.

2.5 Kriptoanaliz

Kriptoanaliz kriptografinin önemli bölümlerinden biri tasarlanmış olan şifreleme algoritmaların ve bu algoritmaların protoköllerinin güvenliğni ölçme işlemini yapar. Kriptoanalizlin temel amacı şifreleme algoritmalarının vaat edilen güvenliğini analiz edilmesi ve var ola bilecek zayıflıkları tesbit edilmesinden ibaretdir. Algoritmadaki zayıflıklar tesbit edildikce yeni tasarlanıcak algoritmalar bu noktalara dikkat edilerek tasarlanır. Bir şifreleme algoritması analiz edilirgen gözlemlenmiş saldırı modelleri ile güvenliğinin

derecesi ölçülür. Bu saldırı modellerinin tasarımda genel ortak varsayım olarak saldırganın sınırsız süre ve kaynağa sahib olması ele alınır.

2.5.1 Kriptoanaliz Teknikleri

Modern şifreleme algoritmasının gelişimi modern kriptoanaliz tekniklerinin gelişimine sebep olmuştur. Geliştirilmiş olan bu kriptoanaliz teknikleri, şifreleme algoritmasının tasarımında bir standart haline gelmiştir. Bu standardların içerisinde tasarlanmış olan algoritmalar saldırılara karşı dayanıklı oldukları varsayılmaktadır. Bu kriptoanaliz teknikleri aşağıda gösterilmiştir.

- Farksal kriptoanaliz
- Doğrusal kriptoanaliz
- Farksal-doğrusal kriptoanaliz
- Ortada buluşma tipi kriptoanalizler

Farksal kirptoanaliz- 1990 senesinde DES algoritmasının analizi için Biham ve Shamir tarafından uygulanmış kriptoanalizdir. Bu teknik en sık kullanılan istatiksel analiz tekniğidir. Bu teknikde doğrusal olmayan algoritmalara, farklı girdiler girilerek algoritmanın üretdiği çıktılar elde edilerek fark dağılım Tablosu elde edilir ve bu Tablolar üzerinden algoritmanın kendi içerisinde barındırdığı karıştırma fonksiyonlarının olasılılığı ölçülür, dolayısıyla farklı olan orijinal metin karakterleri ile şifreli mesaj karakterleri karşılaştırılarak şifreleme fonksiyonlarının formülü elde edilir. Elde edilen formüllerin tek tur için geçerli olduğu farzedilir bu turların bir kaçı analiz edilerek ilgili tur için üretilen anahtarın fonksiyon formülü tasarlanır. Varsayım üzerine elde edilen ilgili tur anahatları ile anahtar üretimi algoritması düzenlenerek şifreleme işleminde kullanılan anahtarlar elde edilir. Anahtarların elde edilmesi sonucunda şifreli mesajı elde eden saldırganlar bu mesajı deşifreleme işlemine sokarak orijinal mesajı elde eder.

Doğrusal kriptoanaliz-En fazla kullanılan kriptoanalizlerden biridir ve 1990 senesinde Matsuri tarafından DES algoritmasına uygulanmıştır. Bu analizde farksal analiz gibi istatiksel analiz tekniğidir. Algoritmada kullanılan doğrusal olmayan yapılar için doğrusal yaklaşım Tablosu yapılır ki bu Tabloda girdi ve çıktı bitlerin doğrusal ifadeleri ve sapmaları yer alır. Bu analizde algoritmadaki doğrusal yapılarda dikkate alınarak girdi ve çıktı bitler arasında doğrusal bir

denklem tasarlanarak şifrelenmiş mesajdan gizli anahtarlar hesaplanarak elde edilir. Anahtar ele geçirme saldırılarında anahtar üreten algoritmalarının tamamı değil birkaç tur için anahtar üreten parçaları ele alınır.

Farksal Doğrusal kriptoanaliz-bu teknikde adından da belli olduğu üzere farksal ve doğrusal teknikler kullanılır. Bu teknik 1994 yılında Langford ve Hellman tarafından geliştirilmiştir. Bu analizde algoritma öncelikle iki parçaya olarak ele alınır. İlk parça için farksal analiz yapılıyor iken ikinci parça için doğrusal kriptoanaliz yapılmaktadır. Özel durumlarda dolayısı ile birinci parçadaki farksal analizin yetersiz kalındığında analiz işlemi olasılılıklar üzerine devam eder.

Ortada buluşma kriptoanaliz-1977 yılında Whitfield Diffie ve Martin Hellman tarafından DES algoritmasının basamaklarını incelemek için geliştirilmiş kriptoanalizdir. Bu analizde iki tane farklı 56 bitlik K_1 ve K_2 anahtarları seçilir. Şifreli mesaj C ise $E_{K_2}(E_{K_1}(P))$ şeklinde hesaplanarak elde edilmiştir. Bir sonraki adım orijinal metin olan P ile $E_{K_1}(P)$ işlemine tabi tutarak elde ettiğimiz değer hem M ara değeri olarak hem de C şifreli metin olarak elde edilir ki bir sonraki adımda C değerini K_2 anahtarı ile $E_{K_2}^{-1}(C)$ deşifreleme işlemine tabi tutulur. Bu işlemin sonucunda elde edilen ara değer ise M' olarak ele alınır. Son olarak elde edilen M ve M' değerleri karşılaştırılarak devam edilirse en kötü ihtimal olarak anahtarın buluna bilme ihtimali 2^{57} en iyi ihtimal değer ise 2^{113}'dür.

3 SOCKET KAVRAMI

İnternet kavramı ilk defa UNIX işletim sisteminin piyasaya çıkışı ile ortaya çıkmıştır, dolayısıyla modern ağ programlamaya dayalı bütün programlar UNIX işletim sisteminin ağ modelinden örnek alınmıştır. Bu nedenle internet kavramı UNIX işletim sistemli bilgisayarların kendi aralarında veri iletimini sağlayan bir araç olmuştur. Bilgisayarlar arası veri iletiminin sağlanması için gereken bağlantıları sağlayan mekanizma Socket olarak bilinmektedir. Bağlantıların sağlanma şekli olarak Socketler iki türlü olmaktadır.

- Akış Socketi (Stream Socket)
- Datagram Socketi

Günümüz teknolojiler daha geliştirilmiş programlama sistemlerinden faydalanmaktadır. Son Geliştirilmiş programlama teknikleri Socket programa mantığının üstünden geliştirilmiştir.

3.1 Akış Socketleri

Akış Socketleri sürekli iki taraflı iletişimi sağlamaktadır, bir başka deyişle iletişim bağlantısı, her iki tarafın iletişimi kesdiği anda iletişim kanalı kapatılır. Akış Socketleri veri iletimi esnasın olası hatalara karşı daha garantilidir. Telefon ile iletişim esnasında kurulan bağlantıyı Akış Socketleri sağlamaktadır. Bu socketlerin performans olarak yavaş çalışması nedeni ise gönderilecek verinin bütünlüğünü, yani verinin bit karşılığındaki sıranın korunmasıdır. Veri iletiminde verinin iletme şeklini sağlayan mekanizmalar ise protokollerdir. Karşılıklı veri iletişimini sağlamak isteyen bilgisayar karşılıklı olarak standartlaştırılmış kurallara uyum sağlamalıdır ki bu kural standartları protokol olarak bilinmektedir. Bu protokollerin bir başka tanımı ise ağ protokolleridir. İletişime geçmek isteyen her hangi bilgisayarlar, sunucular, yönlendiricilerin iletişimi belirlenmiş protokoller üzerinden başlatılmakdadır. Akış Socketlerinin sağlanması için gereken protokol ise TCP (Transfer Control Protocol- Aktarım

kontrol protokolü) protokolüdür. Veri iletimi işlemimini yaparken Akış Socketlerinin verinin bütünlüğünün korunmasını sağlayan mekanizmadır TCP.

3.2 Datagram Socketleri

Datagram Socketi – bağlantısız Socket olarakda bilinmektedir. Datagram Socketler verileri paketler haline getirir ve paketlenmiş veri hazırlandıktan sonra bağlantıyı sağlayıp paketleri gönderdikten sonra bağlantıyı keser. Bu Socketler Akış Socketlerine nazaran daha hızlı çalışmaktadır fakat veri bütünlüğünün garantisi açısından Akış Socketleri gibi performans sağlamazlar. Datagram Socket hücresel veri üzerinden mesaj gönderme işleminde kullanılmaktadır. Bu Socketlerin hızlılık nedeni veri gönderme işleminde verinin korunması ve bütünlüğünün sağlanması gibi işlemleri kontrol etmemesidir. Datagram Socketlerini hızlandıran mekanizma ise onun kullandığı protokoldur, ki bu UDP (User Datagram Protocol) protokoludur. UDP protokolleri TCP protokolleri ile karşılaştırıldığında paketleri sıralı göndermeyi garantilemezler, ki TCP protokollerinin başlık bilgisinde verinin sıra numarasını barındırmaktadır. UDP protokolleri verinin karşı tarafa ulaşıp ulaşmadığını kontrol etmez TCP protokollerinde ise veri karşı tarafa ulaşdığında geriye verinin ulaşdığına dair bilgi gönderir. Veri ulaşmadığı takdirde aynı veriyi tekrar tekrar göndermeyi yapar. UDP protokollerinde ise veri karşı tarafa ulaştıkdan sonra geriye ulaşdığına dair bilgi gönderir fakat, veri ulaşmazsa tekrar göndermeyi denemez.

3.3 Port Kavramı

İki bilgisayar arasında karşılıklı veri iletişimi için, her bilgisayar aynı zamanda hem sunucu olarak hemde istemci olarak faaliyet göstererekde sağlana bilinir. Sunucu olarak faaliyet gösteren bilgisayar Dinleyici aracı ile, belirlenmiş Socketi dinlemek için Socket içerisinde bulunacak veriyi numaralanmış portu dinleyerek gelen veriyi elde eder. Bu nedenle gelen veriler kendisi ile beraberinde girilecek olan portun numarasınıda taşırlar. Modern işletim sistemlerinde port numaralandırılması 0 ile 65535 arasında olur. Numaralandırılmış port içerisinden gelen veri Akış okuyucuları ile çözümlenerek veriyi anlaşılır hale getirir. Sunucu olarak faaliyet gösteren

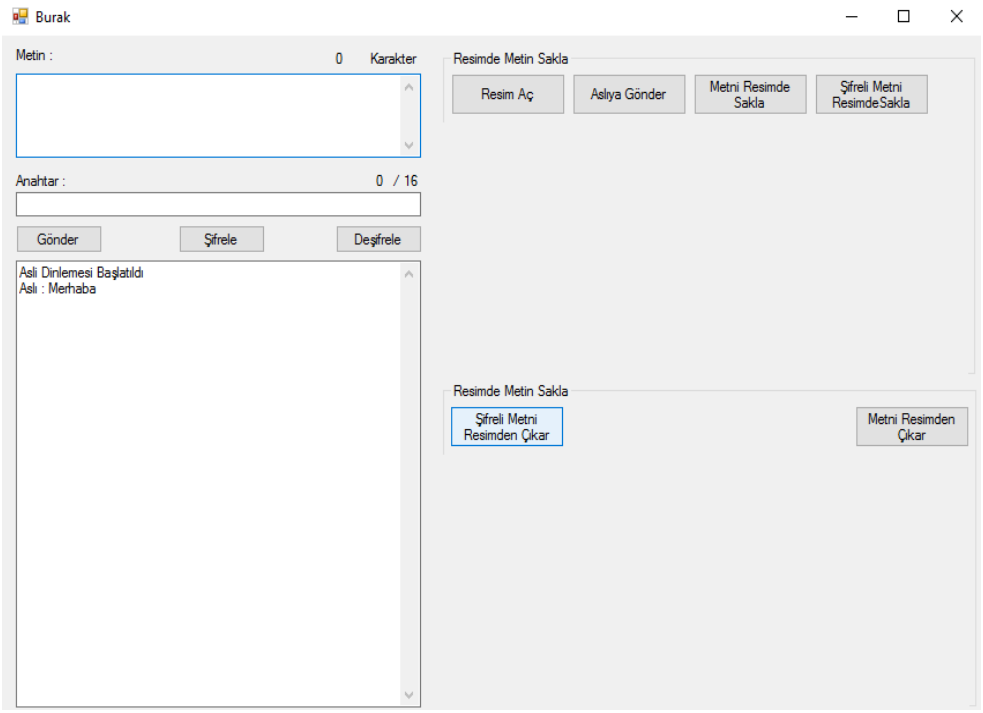
bilgisayar belirlenmiş portu sonsuz bir döngü ile dinlemekte olucaktır. Sonsuz döngüler ile belirlenmiş portu dinlemesi veri akışının Akış Socketi ile sağlanması nedeni ile olmaktadır.

İstemci olarak faaliyet gösteren bilgisayarlar ise sunucu olarak faaliyet gösteren bilgisayarlara veri göndermeden önce sunucunun IP numarası ve gönderilecek verinin hangi port üzerinden kabul edilecekse o portun numarasını gerektirir. Bu bilgiler istemci bilgisayarlara girildikten sonra veriyi ikili sisteme çevirerek aktarma işlemi gerçekleştirilir.

4 SOCKETLER ÜZERİNDEN HABERLEŞMEDE KRİPTOLOJİ METOD UYGULANARAK İLETİŞİMİN SAĞLANMASI

Tez kapsamında yapılan projede Aslı ve Burak isminde karakterler kendi aralarında sohbet ederken ortadaki adam pasiv saldırganın saldırısına maruz kalırlar. Bu projede pasiv saldırgan her iki tarafın mesajlarını dinlemekte fakat müdahele etmemektedir. Yapılan Kriptografik tekniklerle, uygulamalarda yapılan sohbetin ortadaki adam olan Egenin sohbetin içeriğinden haberdar olmamasını sağlamaktır. Kriptoloji araştırmalara göre şifreleme sisteminin tasarımı esnasında saldırgan çok güçlü donatılmış bilgisayara ve sonsuz zamana sahib olduğu var sayılır. Bu araştırmalar nedeni ile benim bu projedeki katkım saldırganın işini dahada zorlaştırarak yani kriptoloji metodların kullanımından ziade gönderilecek olan mesaji Steganografi tekniği ile Resimde gizletilerek karşı tarafa iletilir karşı taraf elde ettiği resmden şifrelenmiş olan gizli mesajı alır ve deşifreleme yöntemi ile orijinal mesajı elde eder. Yapılan uygulamada gizli anahtarlı şifreleme yöntemi olan AES (Advanced Encription Standart), Rijndael algoritması ile de bilinen şifreleme yöntemi kullanılmıştır. Yapılan araştırmalara göre gizli anahtarlı şifreleme yöntemlerinde şifrelenmiş mesajı güvensiz kanal üzerinden iletilirken bu mesajı deşifrelemek için gereken gizli anahtarda güvenli kanal üzerinden gönderilmesi şartdır. Yapılan uygulamada güvenli kanalın olmadığı takdirde şifreli mesajın deşifrelenmesi için gereken anahtar da Steganografi tekniği ile resimde saklanarak karşı tarafa iletilir, ki bu teknik sayesinde yapılacak iletişimde güvenli kanala ihtiyaç duyulmayacaktır.

Yapılan uygulamada ortadaki adam saldırısı yapay olarak gerçekleştirilmiştir ki, Aslı ve Burak Mesajı gönderme işlemi yaparken aynı zamanda Egeyede göndermektedir. Şekil 4.1 de Burak karakterinin kullanacağı uygulamanın arayüzü gösterilmektedir.



Şekil 4.1: Burak Kullanıcısının Uygulama Arayüzü.

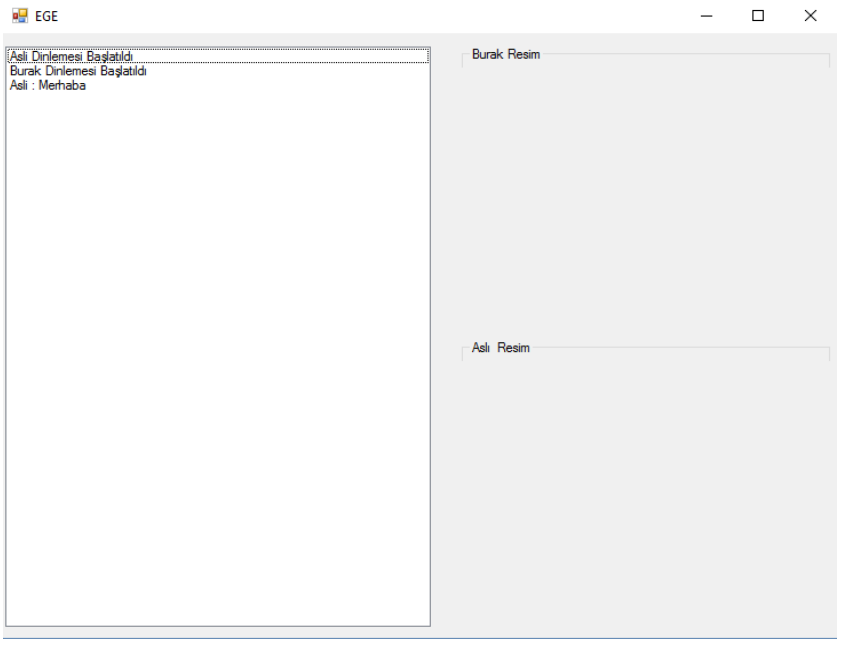
Şekil 4.1'den de görüldüğü gibi Burak Uygulamasının arayüzün yüklenme esnasında gerken kod ilk önce Aslı uygulamasından gelen mesajı elde eden kodlardır. Mesajın elde etmek için Akış Socketlerin çalışma prensibi olarak Aslı ve Burak uygulamaları aynı anda iletişim sağlamalıdır. Burak uygulaması yüklendiğinde hem Aslıdan gelen mesajı almalı ve gönderilecek olan mesaj için bu işlemler aynı anda çalışmalıdır. Burak Aslıdan gelen mesajı almak için Şekil 4.2 gösterilen kodu uygulamanın yüklenme anında çalıştırmalıdır.

```
public TcpClient AsliIstemci;
public NetworkStream AsliAkim;
public StreamReader AsliOkuyucu;

public TcpListener AsliDinleyici;
public Socket AsliSocket;
public string AsliText;

1 reference
public void Asli_Dinleyen()
{
    AsliDinleyici = new TcpListener(3214);
    AsliDinleyici.Start();
    txtEsas.Text += "Asli Dinlemesi Başlatıldı" + Environment.NewLine;
    AsliSocket = AsliDinleyici.AcceptSocket();


    if (!AsliSocket.Connected)
    {
        txtEsas.Text += " Aslıyı Dinlemede bir sorun oluştu" + Environment.NewLine;
    }
    else
    {
        while (true)
        {
            AsliAkim = new NetworkStream(AsliSocket);

            AsliOkuyucu = new StreamReader(AsliAkim);

            AsliText = AsliOkuyucu.ReadLine();


            if (String.IsNullOrEmpty(AsliText))
            {
                txtEsas.Text += "Aslı : Hattan Ayrıldı !!! " + Environment.NewLine;
                break;
            }
            txtEsas.Text += "Aslı : " + AsliText + Environment.NewLine;

        }


    }

}

 public Thread Paralel_Asli_Dinle;
1 reference
private void Form1_Load(object sender, EventArgs e)
{

    Control.CheckForIllegalCrossThreadCalls = false;
    Paralel_Asli_Dinle = new Thread(Asli_Dinleyen);
    Paralel_Asli_Dinle.Start();
      }
```

Şekil 4.2: Burak Uygulamasının Aslı uygulamasından mesajı almak için kodlar

Şekil 4.2'den görüldüğü üzere Burak Uygulamasının yüklendiği anda Thread klassları aracı ile Asli_Dinleyen metodu çağırmış bulunuyoruz. *Thread* klasını aracı olarak kullanma amacımız ise uygulama yüklendiğinde hem Aslıdan mesaj alma hemde Aslıya mesaj gönderme kodlarını paralel olarak çalıştırmamızdır. *Asli_Dinleyen* metodu tetiklendiği anda Burak kendi bilgisayarında 3214 numaralı Sanal Portu *TcpListener* klassı aracı ile dinlemektedir, sonra *AsliDinleyici.Start()* komutu ile dinlemeyi başlatılır. Aslı göndereceği mesajı

NetworkStream Tipine çevirir ve *StreamWriter* klasının aracılığı ile Socket üzerinden mesaj göndermeyi gerçekleştirir. Gönderilen mesajı Socketden almak için dinleyici vasıtası ile yani *AsliSocket = AsliDinleyici.AcceptSocket()* kodunu kullanır. Eğer Aslı Sockete bağlanmada sorun yaşıyor ise Burak kullanıcısına "Aslıyı dinlemede sorun oluştu" mesajını verecektir. Aslı Sockete bağlanmada Sorun yaşamıyor ise sonsuz döngü aracılığı ile Socketin içerisindeki *NetworkStream* tipinden olan veriyi AsliAkim değişkeni içerisine atamasını yapılır. Atama işleminden sonra AsliAkim'in içerisinde olan veriyi *StreamReader* klası aracılığı ile gelen veriyi *StreamReader* tipinden olan AsliOkuyucu değişkenine atıyoruz. Bir Sonraki aşamada *AsliText* değişkenini içeriğini kontrol edeceğiz bağlantı kesildiyi anda bu değişkenin içeriği tamamen boşalıcak ki bu işlem bağlantı kesildiği anda yaranacaktır. Bu değişken içerisinde olan veriyi *string* tipinden olan AsliText değişkenine *AsliOkuyucu.ReadLine()* kodu aracılığı ile gelen veriyi okunur halde atanır. Sonra değişken içerisindeki veriyi txtEsas kontrolüne koyuyoruz. Böylece Aslıdan gelen veriyi elde etmiş olacağız. Bu işlemlerin aynısını Aslı uygulamasında gerçekleşirerek Buraktan gelen veri alına bilinecektir. Aslıdan gelen veriyi Ortadaki adam saldırısını gerçekleştiren Ege, Burak uygulamasında yazılan kod tekniği ile mesajı almaktadır ki Ege uygulamasının arayüzü Şekil 4.3' de gösterilmiştir.



Şekil 4.3 : Ege uygulamasının Arayüzü.

Şekil 4.3'den de görüldüğü üzere Aslının gönderdiği mesaj aynı zamanda Ege uygulamasınada gönderilmiştir.

Aslıdan gelen mesaja karşılık Burak mesaj göndermek istediği taktirde metin alanına mesajını yazar ve Gönder düymesine tıklayarak Şekil 4.4 kodu çalıştırır.

```
private void btnGonder_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(txtMetin.Text))
    {
        MessageBox.Show("Lutfen bir yazı giriniz...");
    }
    else
    {
        Burak_Yazici.WriteLine(txtMetin.Text);
        Burak_Yazici.Flush();

        txtEsas.Text += "Burak : " + txtMetin.Text + Environment.NewLine;

        Ege_Yazici.WriteLine(txtMetin.Text);
        Ege_Yazici.Flush();

        txtMetin.Text = "";
    }
}


public TcpClient Burak_Istemci;
public StreamWriter Burak_Yazici;
public StreamReader Burak_Okuyucu;
public NetworkStream Burak_AgAkim;

public TcpClient Ege_Istemci;
public StreamWriter Ege_Yazici;
public StreamReader Ege_Okuyucu;
public NetworkStream Ege_AgAkim;
1 reference
public void Istemci_Burak()
{
    Burak_Istemci = new TcpClient("localhost", 1236);
    Burak_AgAkim = Burak_Istemci.GetStream();
    Burak_Yazici = new StreamWriter(Burak_AgAkim);



    Ege_Istemci = new TcpClient("localhost", 777);
    Ege_AgAkim = Ege_Istemci.GetStream();
    Ege_Yazici = new StreamWriter(Ege_AgAkim);

}

public Thread Paralel_Burak_Istemci;

1 reference
private void Form1_Load(object sender, EventArgs e)
{

    Control.CheckForIllegalCrossThreadCalls = false;
    Paralel_Asli_Dinle = new Thread(Asli_Dinleyen);
    Paralel_Asli_Dinle.Start();

    Thread.Sleep(4000);
    Paralel_Burak_Istemci = new Thread(Istemci_Burak);
    Paralel_Burak_Istemci.Start();
}
```

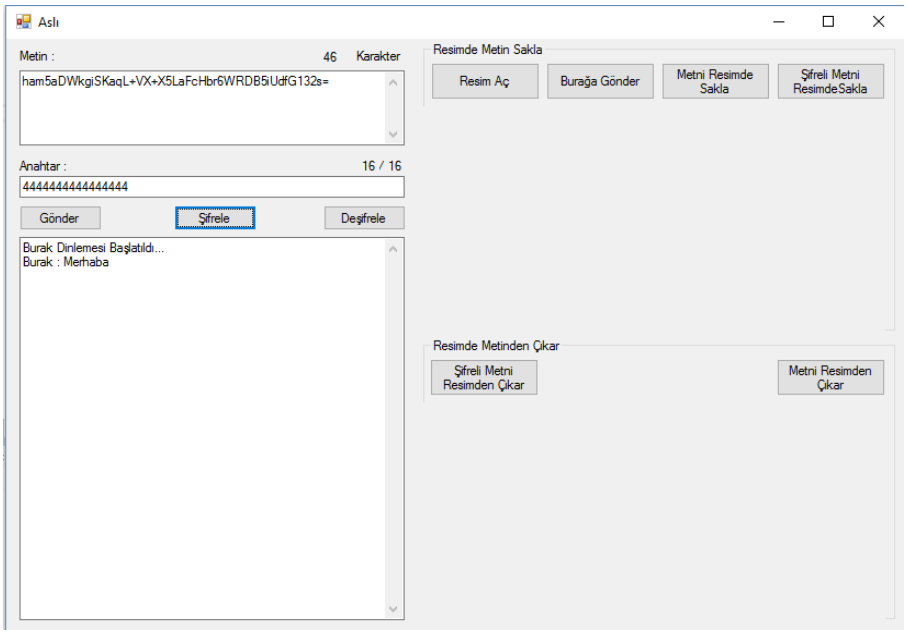
Şekil 4.4 : Mesaj Gönderilmesi için çalıştırılan kod.

Şekil 4.4'den görüldüğü üzere Burak uygulamasının çalıştırıldığı anda paralel çalıştırılması gereken bir diğer kod ise Burağın istemci olarak karşı tarafa bağlanmasıdır. *Thread.Sleep(4000)* kodu aracılığı ile kodun çalışmasını 4 saniye bekletiyoruz ki Aslı uygulaması açıldığı anda Burak Uygulamasını dinlemeye

başlıyacaktır ki bu TCP protoköl gereği İstemci ve dinleyici aynı anda çalıştırılmalıdır. *Control.CheckForIllegalCrossThreadCalls = false* kodu ise Asl_Dinleyen metodu ile İstemci_Burak metodu paralel olarak çalıştırıldığında bu çalışma esnasında herhangi bir çakışmayı engelleyecektir.

Istemci_Burak metodunda ise *TcpClient* tipinden olan Burak_Istemci değişkenine *TcpClient* klassının nesne örneği alınırken bağlanılacak bilgisayara hangi IP numaralı ağdan çıkış yapılacağını ve gönderilecek verinin karşı tarafa hangi port üzerinden iletileceği isteniyor ki bu uygulamalar aynı Bilgisayarda olduğundan IP bölümüne "*localhost*" olarak, port numarasına ise Aslı uygulamasının çalıştığı anda dinlediyi port numarasını yazıyoruz yani *1236* numaralı portdan iletilecektir. Socket üzerinden veri göndermek için gönderilecek veri *NetworkStream* tipinden olmalı ki bu işlemi *Burak_AgAkim = Burak_Istemci.GetStream()* kodu aracılığı ile yapılmaktadır. Gönderilecek veri *NetworStream* tipine çevrildikten sonra veriyi göndermek için *Burak_Yazici = new StreamWriter(Burak_AgAkim)* kodu kullanılmaktadır. Gönderilmiş olan mesajı Ege uygulamasına göndermek içinde aynı teknik kullanılır. Uygulanan bu teknik Burak uygulamasının çalışma esnasında yapılmaktadır, dolayısıyla Burak uygulaması yüklenirken Aslı uygulaması ve Ege uygulaması ile iletişim sağlanır. İletişim sağlandıktan sonra mesajı karşı tarafa göndermek için Gönder düymesinin tıklandığı anda çalışacan kod ise öncelikle metin kutusunun boş olup olmadığını kontrol ediyoruz eğer metin kutusu boş olursa "*Lütfen bir yazı giriniz…*" diye bir uyarı mesajı kullanıcıya gösterilecektir. Metin kutusu boş olmadığı taktirde ise mesajı göndermek için aracı olan *Burak_yazici* değişkeninin *WrıteLine metodu* kullanılarak mesaj karşı tarafa gönderilir ve sonrasında bu metodun içerisini *Flush()* metodu ile *Burak_yazici* değişkeninin içini boşaltıyoruz. Aynı teknik ile mesaj Ege uygulamasına da gönderilmiş olucaktır.

Burak mesajı gönderdiği taktirde aynı mesaj pasiv saldırgan Egeyede ulaşıldığı için Aslı göndereceği cevabı Rijndeal şifreleme algoritması ile şifreleyerek gönderir ki, bu işlem Şekil 4.5'de gösterilmiştir .



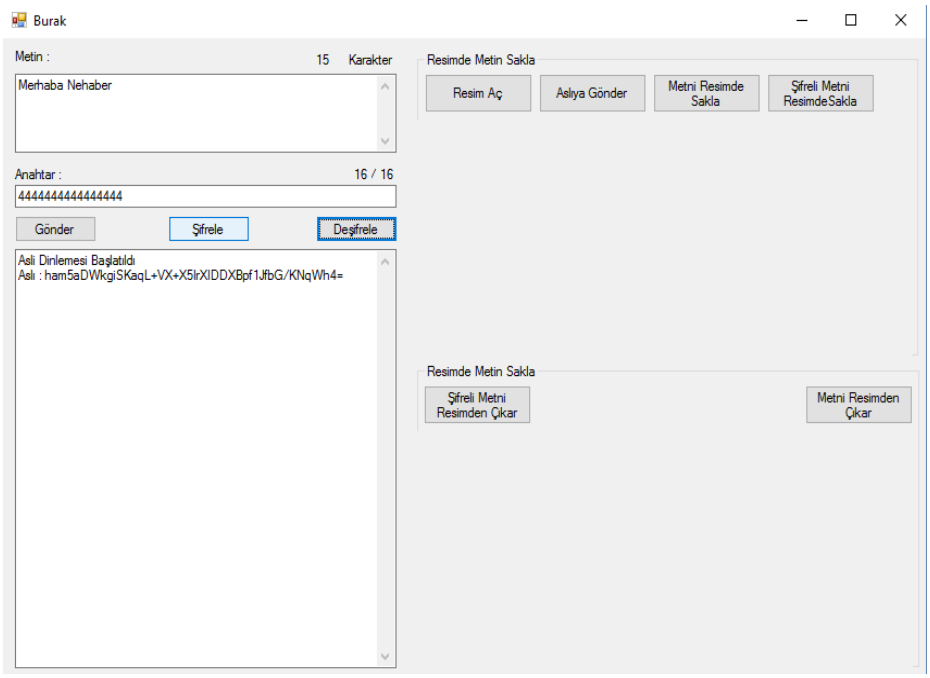
Şekil 4.5: Şifrelenen Mesaj.

Şekil 4.5'de görüldüğü üzere Aslı mesajı şifrelemek için gereken metni metin kutusuna yazar ve şifrelemek için gereken onaltı karakterlik gizli anahtarı girer ve şifrele düymesine tıklandığında çalışan kod Şekil 4.6'da gösterilmiştir.

```
public static string Metin { get; set; }

private static string Anahtar;
private static string BV;
0 references
private void btnSifrele_Click(object sender, EventArgs e)
{
    if (txtAnahtar.Text.Length == 16)
    {
        txtMetin.Text = Sifrele(txtMetin.Text)
    }
    else
    {
        MessageBox.Show("Anahtar Karakterlerinin sayısı Yetersiz Kaldı lütfen 16'a tamamlayın", "Uyari");
    }
}

1 reference
private static string Sifrele(string metin)
{
    byte[] AcikMetinByte = System.Text.ASCIIEncoding.Unicode.GetBytes(metin);
    AesCryptoServiceProvider aes = new AesCryptoServiceProvider();
    aes.BlockSize = 128;
    aes.KeySize = 128;
    aes.Key = System.Text.ASCIIEncoding.ASCII.GetBytes(Anahtar);
    aes.IV = System.Text.ASCIIEncoding.ASCII.GetBytes(BV);
    aes.Padding = PaddingMode.Zeros;
    aes.Mode = CipherMode.CBC;
    ICryptoTransform kripto = aes.CreateEncryptor(aes.Key, aes.IV);
    byte[] Sifrelenmis = kripto.TransformFinalBlock(AcikMetinByte, 0, AcikMetinByte.Length);
    kripto.Dispose();
    return Convert.ToBase64String(Sifrelenmis);
}
```

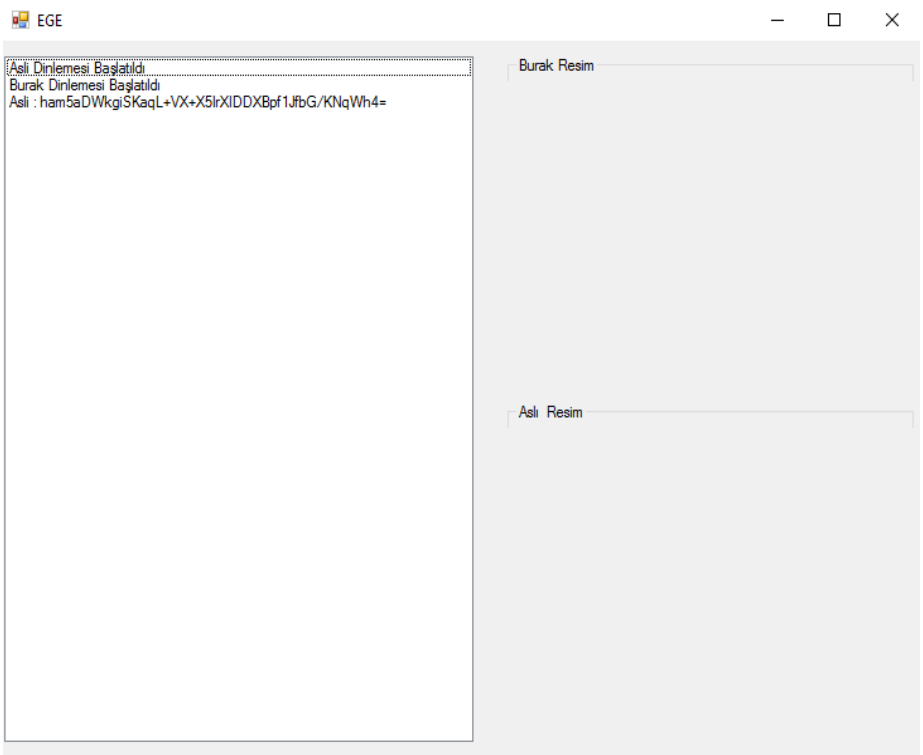
Şekil 4.6: AES şifreleme algoritması.

Aslı şifrelemek isteyeceği metni ve gizli anahtarı girdikten sonra Şifrele düymesine tıkladığında öncelikle gizli anahtarın 16 karaktere eşit olup olmadığını kontrol eder eğer 16 karaktere eşit olmazsa kullanıcıya "*Anahtar Karakterlerinin sayısı Yetersiz Kaldı lütfen 16'a tamamlayın*" uyarı mesajını verecektir. Girilen anahtarın uzunluğu 16 karaktere eşit ise *Sifrele()* metodunu çağırır. Bu metod çağırılırken parametre olarak *string* tipinden *metin* değişkenini gerektirir ki bu da şifrelenecek metindir. Bu metod çalıştıkda şifrelenecek metnin karakterlerinin *ASCII* değerlerini alarak *byte* tipinde bir dizisi yaratır. AES şifreleme işlemini yapan sınıf ise *AesCryptoServiceProvider* sınıfıdır. Bu sınıfın nesne örneğini aldıktan sonra AES algoritmasının blok boyutu 128 olarak belirlenir. Sonra ise şifreleyecek olan anahtarın boyutu 128 olarak belirlenir. Sonra girilmiş olan anahtarında ASCII değerleri alınarak ele alınır. Sonraki adımda Başlangıç vektörü belirlenir ki bu AES algoritmasının Şifre Blok Zincirlemesi (Cipher Block Chaining Mode - CBC) modunda çalışması için gereklidir. Bir sonraki aşamada ise eğer algoritmanın blok boyutu 128 bitden düşük olursa gerekli girilmiş olan verinin bit karşılığını 128 bite "0" tamamlar. Sonraki aşama ise AES algoritmasının çalışma modunu CBC olarak seçme aşamasıdır. Bir sonraki aşamada AES algoritması başlamadan önce her tur için üretilecek anahtar *ICryptoTransform kripto = aes.CreateEncryptor(aes.Key, aes.IV)* kodu ile yapılmaktadır. Anahtar üretiminden hemen sonra şifreleme işlemi *kripto.TransformFinalBlock(AcikMetinByte, 0, AcikMetinByte.Length)* kodu ile yapılıp, adı *Sifrelenmis* olan byte dizisine atanır. Atama işleminden sonra AES algoritmasının bloklarını *kripto.Dispose();* kodu ile temizlenir. Şifrelenmiş metin byte dizisine atandıktan sonra geriye şifrelenmiş metini döndürmek için *return Convert.ToBase64String(Sifrelenmis)* kodu çalışacaktır. Şifreleme işleminden sonra şifrelenmiş metin, metin kutusunda gösterilmiş olacak ki bu metni karşı tarafa gönderilir. Şifrelenmiş metin gönderildikten sonra gizli anahtar güvenli kanal üzerinden deşifrelemek için gönderilir ki Şekil 4.7 de gösterilmiştir.



Şekil 4.7: Şifrelenmiş metnin deşifrelenmesi

Şekil 4.7'den de görüldüğü üzere Aslının gönderdiği şifrelenmiş metin Burağa iletilmiş bulunmaktadır. Bu şifrelenmiş metin pasiv saldırgan olan Egeyede iletilmiştir (Şekil 4.8).



Şekil 4.8: Ege uygulaması Şifrelenmiş metin.

Şekil 4.8'den de görüldüğü üzere Aslının göndermiş olduğu şifreli metin Egeyede ulaşmaktadır. Egede gizli anahtar olmadığından dolayı bu metni deşifreleyemeyecektir.

Burak aldığı şifreli metin ve gizli anahtar ile Deşifrele düymesine tıkladığı anda çalışacak kod Şekil 4.9'da gösterilmiştir.

```
private void btnDesifrele_Click(object sender, EventArgs e)
{
    if (txtAnahtar.Text.Length == 16)
    {
        txtMetin.Text = DeSifrele(txtMetin.Text);
    }
    else
    {
        MessageBox.Show("Anahtar Karakterlerinin sayısı Yetersiz Kaldı lütfen 16'a tamamlayın", "Uyari");
    }
}
1 reference
public static string DeSifrele(string SifrelenmisMetin)
{
    byte[] sifrelenmisMbyte = Convert.FromBase64String(SifrelenmisMetin);
    AesCryptoServiceProvider aes = new AesCryptoServiceProvider();
    aes.BlockSize = 128;
    aes.KeySize = 128;
    aes.Key = System.Text.ASCIIEncoding.ASCII.GetBytes(Anahtar);
    aes.IV = System.Text.ASCIIEncoding.ASCII.GetBytes(BV);
    aes.Padding = PaddingMode.Zeros;
    aes.Mode = CipherMode.CBC;
    ICryptoTransform kripto = aes.CreateDecryptor(aes.Key, aes.IV);
    byte[] gizli = kripto.TransformFinalBlock(sifrelenmisMbyte, 0, sifrelenmisMbyte.Length);
    kripto.Dispose();
    return System.Text.ASCIIEncoding.Unicode.GetString(gizli);
}
```

Şekil 4.9: AES algoritmasının deşifrelenmesi

Şifrelenmiş metni elde eden Burak bu metni metin kutusuna yerleştirip Deşifrele düymesine tıkladığı anda öncelikle gizli anahtarın boyutunun 16 karaktere eşit olup olmadığını kontrol edilir. Eğer yanlışlıkla anahtar boyutu 16 karakterden farklı olursa kullanıcıya "*Anahtar Karakterlerinin sayısı Yetersiz Kaldı lütgen 16'a tamamlayın*" mesajı gösterilecektir. Aksi taktirde Deşifrele metodu çağrılacaktır ki bu metod parametre olara *string* tipinden *SifrelenmisMetin* istemektedir. Şifrelenmiş metin elde edildikten sonra bu metin *byte* tipinden bir diziye atanır. Sonra AES algoritmasının gereken parametrelerini girmek için *AesCryptoServiceProvider* klassının nesne örneği alınır. Bir sonraki aşama ise şifrelenmiş metnin algoritmasının blok boyutu, ve gizli anahtarın boyutu girilir. Bir sonraki aşamada girilmiş olan anahtarın ve Başlangıç vektörün ASCII değerleri girilir. Bir sonraki aşamada dolgulama işleminin sıfırlarla yapılacağı bildirilir. Deşifreleme algoritmasının Şifre Blok Zincirlemesi (Cipher Block Chaining Mode - CBC) blok modu ile çalışacağı bildirilir. Bir sonraki aşamada döngü anahtarlarının üretimi için gizli anahtar ve başlangıç vektörü girilerek

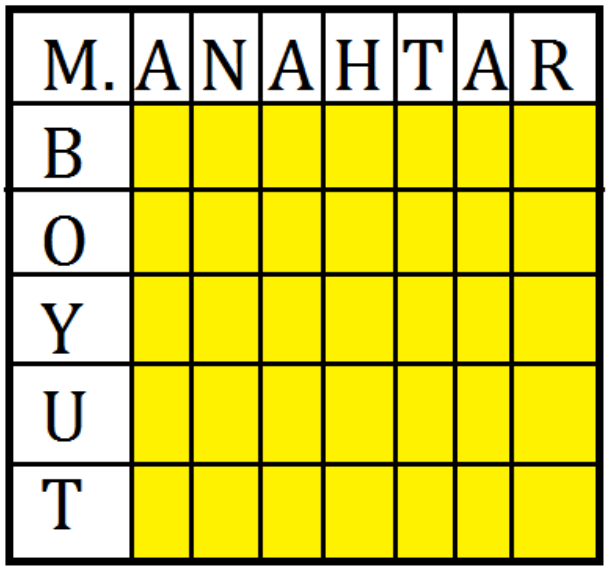
işlem başlatılır. Anahtar üretiminden sonra deşifreleme *kripto. TransformFinalBlock(sifrelenmisMbyte,0,sifrelenmisMbyte.Length)* kodu ile yapılarak *byte* tipinden bir diziye atanır. Bir sonraki aşamada AES deşifreleme algoritmasının blokları boşaltılır. Sonraki aşamada ise elde edilmiş byte değerlerin ASCII kodları ve sonrasında bu kodların karakter karşılığı üretilerek geriye dönderilir ki bu karakterle metin kutusunda gösterilir. Bu şekilde iletişimin sağlanması takdirde ortadaki odam saldırganı olan Ege bu iletişimin şifreli olduğunu anlayıp bu belli bir kriptoanaliz yaparak bu anahtarı bulma çabasında bulunacaktır. Hem böyle bir ihtimale el vermemek adına hemde gizli anahtarı güvenli kanal olmadığı takdirde karşı tarafa anahtarı güvensiz kanal üzerinden "png" formatlı resimde hem şifreli mesaj hemde gizli anahtar gizletilerek karşı tarafa gönderilebilinir.

Burak gelen şifreli mesaja karşılık olarak vereceği cevabı şifreleyerek Resimde gizletmek için önce Resim Aç düymesine tıklayarak "png" formatlı resmi seçerken arka tarafda çalışan kof Şekil 4.10 da gösterilmiştir.

```
Image File1;
OpenFileDialog f;
1 reference
private void button2_Click(object sender, EventArgs e)
{
    f = new OpenFileDialog();
    f.Filter = "PNG|*.png";


    if (f.ShowDialog() == DialogResult.OK)
    {
        File1 = Image.FromFile(f.FileName);
        pcbGonderilen.Image = File1;

        Bitmap rsm = new Bitmap(pcbGonderilen.Image);

        Bitmap yrsm = new Bitmap(rsm.Width + 1, rsm.Height + 1);

        for (int i = 0; i < yrsm.Height; i++)
        {
            for (int j = 0; j < yrsm.Width; j++)
            {
                yrsm.SetPixel(j, i, Color.FromArgb(255, 255, 255, 255));
            }
        }

        Graphics g = Graphics.FromImage(yrsm);

        g.DrawImage(rsm, new Point(1, 1));

        Random rnd = new Random();
        int resimAdi = rnd.Next();
        string CariProjeAdres = Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory.GetCurrentDirectory()));

        yrsm.Save(CariProjeAdres + "\\EklenenResimler\\" + "Buyutulen" + resimAdi.ToString() + ".png");
        pcbGonderilen.ImageLocation = CariProjeAdres + "\\EklenenResimler\\" + "Buyutulen" + resimAdi.ToString() + ".png";

    }

}
```

Şekil 4.10: Resim açma işlemi için çalışan kod.

Seçilecek resimde şifreli metni ve gizli anahtarı saklamak için seçilmiş olan resmin genişliği 1 piksel genişletilir ki burada saklanılacak olan metnin boyutu saklanır. Gizli anahtarı saklamak için ise resmin yüksekliği 1 piksel çoğaltılır. Şekil 4.11 de saklanılacak olan metin, bu metnin boyutu, ve gizli anahtarın saklanılacağı pikseller gösterilmiştir.



Şekil 4.11: Şifreli Metnin Resimde Saklanması

Şekil 4.11'den de görüldü üzere Resmin orijinal boyutu sarıya boyanmış alan kadardır. Resim Aç düymesi tıklandığında yüksekliğe ve genişliğe 1 piksel eklenerek düzenlenmiş resmin beyaz yükseklik olan alanına sarı alana saklanılacak Şifrelenmiş metnin boyutu, beyaz olan genişlik alanına ise gizli anahtar saklanır. Bu işlemi yapmak için öncelikle *OpenFileDialog* klasının nesne örneği alınır ve bu nesne örneğine *f* adı konulur. Sonra f değişkeninin *Filter* özelliğine "*png*" değerini atayarak açılacak pencerede sadece "png" uzantılı resimler gösterilmesi sağlanılır. Açılan pencerede "png" formatlı resmim seçtikten sonra "OK" düymesine tıklandıktan sonra seçilen Resim *Image* tipinden *File1* değişkenine *f.FileName* aracılığı ile atanır. *File1* içerisindeki Resim *pcbGonderilen* kontrolünün *Image* özelliğine atanır. Sonra resimleri bit cinsinde tutan klas *Bitmap* klasının nesne örneği alınır bu nesne örneği *rsm* olarak belirlenir ve içerisinde *pcbGonderilen* kontrolünün içerisindeki resim atanır. Sonra ise başka bir *Bitmap* klasının nesne örneği alınır ki bu da *yrsm* olarak belirlenir. Bu işlem aracılığı ile yapay resim yapılır ve bu yapay resmin genişliği seçilmiş resmin genişliğinden 1 piksel, yüksekliği ise

seçilmiş resmin yüksekliğinden 1 piksel fazla olarak belirlenir. Bir sonraki aşamada yapay düzeltilmiş resmin piksellerini beyaz renge boyamak için yapay resmin piksellerinde döngü ile dönülür birinci döngü resmin yüksekliğinde dönerken ikinci döngü resmin genişliğinde dönmektedir. Bu döngü içerisinde her pikselin rengleri *yrsm.SetPixel(j, i, Color.FromArgb(255, 255, 255, 255))* kodu aracılığıyla beyaz renge boyanır. Beyaz renge boyama işleminden sonra yapay resim ile seçilmiş olan resmi üstüste yerleştirmek için yapay olan resmi *Graphics* tipinde *g* değişkeni tanımlanarak yapay resim *Graphics.FromImage(yrsm)* kodu ile atanır. Seçilmiş resmi yapay resmin üzerine yerleştirmek için *g* değişkeninin *DrawImage(rsm, new Point(1, 1))* metodu ile yapılır ki böylece resimler üst üste yerleştirilmiş olunur. Bu işlemden sonra düzenlenmiş olan resmin adını rastgele isim koymak için *Random* klasının nesne örneği alınır ve rnd olarak belirlenir. Resim adının belli bir parçası rastgele seçilmiş rakam olması için *rnd.Next()* kodu çalıştırılır. Bir sonraki aşamada Cari Projenin adresini elde etmek için *Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory.GetCurrentDirectory()))* kodu çalıştırılır. Sonra düzenlenmiş olan resmi kaydetmek için *yrsm.Save(CariProjeAdres + "\\EklenenResimler\\" + "Buyutulen" + resimAdi.ToString() + ".png")* kodu çalıştırılır. Düzenlenmiş resmi uygulamada görmek için ise *pcbGonderilen.ImageLocation = CariProjeAdres + "\\EklenenResimler\\" + "Buyutulen" + resimAdi.ToString() + ".png";* kodu çalıştırılır ve arayüzde gözüken resim şifreli metin saklamak için hazır vaziyyetdedir Metin saklamak için düzenlenmiş resimde metin saklamak içim "Şifreli Metni Sakla" düymesine tıklandığında önce saklanılacak metnin boyutu ilave olunmuş yükseklik piksellerinde saklamak için Şekil 4.12 de gösterilen kod çalışacaktır.

```
public void ResimdeSifreliMetinBoyutSakla(int TextBoyut, Bitmap rsm)
{
    int R, G, B;
    int nR, nG, nB;
    int ResminYukseklikPixelSayisi = rsm.Height;
    string TextBoyutbit = StringToBinary(Convert.ToString(TextBoyut));
    if (TextBoyutbit.Length >= ResminYukseklikPixelSayisi * 3)
    {
        MessageBox.Show("Girdiğiniz Metin Resime Sığdırılamıyor", "Uyarı");
    }
    else
    {
        for (int i = 0; i < rsm.Height; i++)
        {
            Color piksel = rsm.GetPixel(0, i);
            R = 252; G = 252; B = 252;
            if (!string.IsNullOrEmpty(TextBoyutbit))
            {
                int SonKarbit1 = Convert.ToInt32(TextBoyutbit.Substring(TextBoyutbit.Length - 1, 1));
                nR = R + SonKarbit1;
                TextBoyutbit = TextBoyutbit.Remove(TextBoyutbit.Length - 1, 1);
                rsm.SetPixel(0, i, Color.FromArgb(nR, G, B));
                R = nR;
                if (string.IsNullOrEmpty(TextBoyutbit))
                {
                    rsm.SetPixel(0, i, Color.FromArgb(nR, piksel.G, piksel.B));
                    break;
                }
            }
            else
            {
                rsm.SetPixel(0, i, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
            }
            if (!string.IsNullOrEmpty(TextBoyutbit))
            {
                int SonKarbit2 = Convert.ToInt32(TextBoyutbit.Substring(TextBoyutbit.Length - 1, 1));
                nG = G + SonKarbit2;
                TextBoyutbit = TextBoyutbit.Remove(TextBoyutbit.Length - 1, 1);
                rsm.SetPixel(0, i, Color.FromArgb(R, nG, B));
                G = nG;
                if (string.IsNullOrEmpty(TextBoyutbit))
                {
                    rsm.SetPixel(0, i, Color.FromArgb(R, nG, piksel.B));
                    break;
                }
                else
                {
                    rsm.SetPixel(0, i, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
                }
                if (!string.IsNullOrEmpty(TextBoyutbit))
                {
                    int SonKarBit3 = Convert.ToInt32(TextBoyutbit.Substring(TextBoyutbit.Length - 1, 1));
                    nB = B + SonKarBit3;
                    TextBoyutbit = TextBoyutbit.Remove(TextBoyutbit.Length - 1, 1);
                    rsm.SetPixel(0, i, Color.FromArgb(R, G, nB));
                    if (string.IsNullOrEmpty(TextBoyutbit))
                    {
                        rsm.SetPixel(0, i, Color.FromArgb(R, G, nB));

                        break;
                    }
                }
                else
                {
                    rsm.SetPixel(0, i, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
                }
            }
            Random rnd = new Random();
            int resimAdi = rnd.Next();
            string CariProjeAdres = Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory.GetCurrentDirectory()));
            pcbGonderilen.Image = rsm;


        }
    }
```

Şekil 4.12: Resimde Metin Boyut Saklama

Şekil 4.12'den de görüldüğü üzere Resimde metin boyut saklamak için öncelikle *int* veri tipinde resmin her piksel Kırmızı (Red), Yeşil (Green), Mavi (Blue) değerlerini tutmak için değişken tanımlanır. Bir sonraki aşamada veri saklandıkdan sonra bu renglerin deyişen değerlerini tutmak için ise *int nR, nG, nB* değişkenleri tanımlanır. Sığdırılmak istenen metnin boyutu resmin yüksekliğini aşarsa metin saklamak için bir başka resim seçilmesi gerekir. Bu işlemi kontrol etmel için her piksel 3 bitlik değer yerleştirilirse öncelikle resmin yükseklik değeri *int ResminYukseklikPixelSayisi = rsm.Height* şeklinde alınır. Resimde veri saklamak için her saklanmak istenen her pikselin RGB değerlerinin her birinde en az öneme sahip olan bit yani RGB değerlerinin bit karşılığındaki ver "0" ile bitiyorsa bu son değerin yerine saklanması gereken verinin biti koyula bilir böylece resmin her pikselinin RGB değeri çok az farkla değiştirelecek fakat içerisinde saklanmak istenen veriyide saklayacaktır. Bu değişikliği insan göz farketmeyecektir. Bunun için saklanma istenen veri bit cinsine *string TextBoyutbit = StringToBinary(Convert.ToString(TextBoyut))* kodu ile çevrilerek string değere atanacaktır. Bir sonraki aşamada resmin yüksekliğindeki piksel sayısının 3 misli saklanmak istenen veriden büyük ise saklama işlemi gerçekleşecektir. Aksi taktirde " *girmek istediğiniz veri Resime sığdırlamıyor*" mesajı gösterilecektir. Metin saklama işleminde resmin yüksekliğindeki bütün bitler teker teker *Color piksel = rsm.GetPixel(0, i)* kod aracılığı ile ele alınır. Bir sonraki aşamada değişken olarak tanımlanan RGB değerlerinin her birine 252 değerini atama nedeni ise sonu çift rakamla biten bir sayının bit karşılığı "0" ile bitmektedir ve bu değerin yerine saklanmak istenen verinin bit cinsinden son bitini koya biliriz. Bir sonraki aşamada girilecek olan veri saklanması bitmişmi bitmemişmi kontrol ediyoruz ki bu kontrolün amacı bir sonraki döngüde saklanmak istenen veri bitmiş ola bilir. Döngünün ilk adımı olduğunu varsayarsak saklanmak istenen veri boş değildir. Bu veri boş olmadığına göre saklanmak istenen verinin son biti

int SonKarbit1 = Convert.ToInt32(TextBoyutbit.Substring(TextBoyutbit.Length - 1, 1));

şeklinde alınır ve bu değer eğer "1" ise pikselin R değeri ile toplanarak yeni oluşacak bit yani nR değeri 253 olacaktır, eğer bu değer "0" olucaksa yine toplama işlemi yapılarak 252 değerini alacaktır. Bit deyişmesi gerçekleştikten

sonra o pikselin R *değerini* *rsm.SetPixel(0, i, Color.FromArgb(nR, G, B))* belirlemiş oluyoruz. Bu işlemden sonra saklanmak istenen veri bitmişse eğer cari pikselin G ve B değeri olduğu gibi bırakılacaktır ve döngünün sonuna gelinecektir. Saklanmak istenen veri bitmemişse eğer bir sonraki reng değeri olan G değeri aynı şekilde değiştirilecektir. Bu işlemden sonra saklanmak istenen veri bitmişse eğer cari pikselin B değeri olduğu gibi atanacaktır. Saklanmak istenen veri bitene kadar bu işlem tekrarlanacaktır. Döngünün sonuna gelindiğinde ise boyut saklanan resim pcbGonderilen kontrolünde gösterilecektir. Bu işlemden sonra sıra gizli anahtarı resmin genişliğinde saklamak olucaktır. Bu işlem Şekil 4.13'de gösterilmiştir.

```
public void ResimdeAnahtarSakla(string Anahtar, Bitmap rsm)
{
    int R, G, B;
    int nR, nG, nB;
    int ResminGenislikPixelSayisi = rsm.Width;
    string Anahtarbit = StringToBinary(Convert.ToString(Anahtar));

    if (Anahtarbit.Length <= ResminGenislikPixelSayisi * 3)
    {
        for (int j = 1; j < rsm.Width; j++)
        {
            Color piksel = rsm.GetPixel(j, 0);
            R = 252; G = 252; B = 252;
            if (!string.IsNullOrEmpty(Anahtarbit))
            {
                int AnSonbit=Convert.ToInt32(Anahtarbit.Substring(Anahtarbit.Length - 1, 1));
                nR = R + AnSonbit;
                Anahtarbit = Anahtarbit.Remove(Anahtarbit.Length - 1, 1);
                rsm.SetPixel(j, 0, Color.FromArgb(nR, G, B));
                R = nR;
                if (string.IsNullOrEmpty(Anahtarbit))
                {
                    rsm.SetPixel(j, 0, Color.FromArgb(nR, piksel.G, piksel.B));
                    break;
                }
                else
                {
                    rsm.SetPixel(j, 0, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
                }
                if (!string.IsNullOrEmpty(Anahtarbit))
                {
                    int AnSonbit2 = Convert.ToInt32(Anahtarbit.Substring(Anahtarbit.Length - 1, 1));
                    nG = G + AnSonbit2;
                    Anahtarbit = Anahtarbit.Remove(Anahtarbit.Length - 1, 1);
                    rsm.SetPixel(j, 0, Color.FromArgb(R, nG, B));
                    G = nG;
                    if (string.IsNullOrEmpty(Anahtarbit))
                    {
                        rsm.SetPixel(j, 0, Color.FromArgb(R, nG, piksel.B));
                        break;
                    }
                }
                else
                {
                    rsm.SetPixel(j, 0, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
                }
                if (!string.IsNullOrEmpty(Anahtarbit))
                {
                    int AnSonbit3 = Convert.ToInt32(Anahtarbit.Substring(Anahtarbit.Length - 1, 1));
                    nB = B + AnSonbit3;
                    Anahtarbit = Anahtarbit.Remove(Anahtarbit.Length - 1, 1);
                    rsm.SetPixel(j, 0, Color.FromArgb(R, G, nB));
                    if (string.IsNullOrEmpty(Anahtarbit))
                    {
                        rsm.SetPixel(j, 0, Color.FromArgb(R, G, nB));

                        break;
                    }
                }
                else
                {
                    rsm.SetPixel(j, 0, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
                }
                Random rnd = new Random();
                int resimAdi = rnd.Next();
                string CariProjeAdres = Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory.GetCurrentDirectory()));
                pcbGonderilen.Image = rsm;
            }
        }
    }
    else
    {
        MessageBox.Show("Şifreli Metin Resime Sığdırılamıyor Lütfen daha Büyük Resim Seçiniz...", "Uyarı");
    }
}
```

Şekil 4.13: Gizli Anahtar Resimde Saklama

Şekil 4.13'den de görüldüğü üzere Resimde Gizli Anahtar saklamak için öncelikle *int* veri tipinde resmin her piksel Kırmızı (Red), Yeşil (Green), Mavi (Blue) değerlerini tutmak için değişken tanımlanır. Bir sonraki aşamada veri

saklandıkdan sonra bu renglerin deyişen değerlerini tutmak için ise *int nR, nG, nB* değişkenleri tanımlanır. Sığdırılmak istenen metnin boyutu resmin genişliğini aşarsa metin saklamak için bir başka resim seçilmesi gerekir. Bu işlemi kontrol etmel için her piksel 3 bitlik değer yerleştirilirse öncelikle resmin genişlik değeri *int ResminGenislikPixelSayisi = rsm.Width* şeklinde alınır. Resimde veri saklamak için her saklanmak istenen her pikselin RGB değerlerinin her birinde en az öneme sahip olan bit yani RGB değerlerinin bit karşılığındaki ver "0" ile bitiyorsa bu son değerin yerine saklanması gereken verinin biti koyula bilir böylece resmin her pikselinin RGB değeri çok az farkla değiştirelecek fakat içerisinde saklanmak istenen veriyide saklayacaktır. Bu değişikliği insan göz farketmeyecektir. Bunun için saklanma istenen veri bit cinsine *string Anahtarbit = StringToBinary(Convert.ToString(Anahtarbit))* kodu ile çevrilerek string değere atanacaktır. Bir sonraki aşamada resmin genişliğindeki piksel sayısının 3 misli saklanmak istenen veriden büyük ise saklama işlemi gerçekleşecektir. Aksi taktirde " *girmek istediğiniz veri Resime sığdırlamıyor*" mesajı gösterilecektir. Metin saklama işleminde resmin genişliğindeki bütün bitler teker teker *Color piksel = rsm.GetPixel(j, 0)* kod aracılığı ile ele alınır. Bir sonraki aşamada değişken olarak tanımlanan RGB değerlerinin her birine 252 değerini atama nedeni ise sonu çift rakamla biten bir sayının bit karşılığı "0" ile bitmektedir ve bu değerin yerine saklanmak istenen verinin bit cinsinden son bitini koya biliriz. Bir sonraki aşamada girilecek olan veri saklanması bitmişmi bitmemişmi kontrol ediyoruz ki bu kontrolün amacı bir sonraki döngüde saklanmak istenen veri bitmiş ola bilir. Döngünün ilk adımı olduğunu varsayarsak saklanmak istenen veri boş değildir. Bu veri boş olmadığına göre saklanmak istenen verinin son biti

int SonKarbit1 = Convert.ToInt32(Anahtarbit.Substring(Anahtarbit.Length - 1, 1));

şeklinde alınır ve bu değer eğer "1" ise pikselin R değeri ile toplanarak yeni oluşacak bit yani nR değeri 253 olacaktır, eğer bu değer "0" olucaksa yine toplama işlemi yapılarak 252 değerini alacaktır. Bit deyişmesi gerçekleştikten sonra o pikselin R *değerini rsm.SetPixel(j, 0, Color.FromArgb(nR, G, B))* belirlemiş oluyoruz. Bu işlemden sonra saklanmak istenen veri bitmişse eğer cari pikselin G ve B değeri olduğu gibi bırakılacaktır ve döngünün sonuna gelinecektir. Saklanmak istenen veri bitmemişse eğer bir sonraki reng değeri olan G değeri aynı şekilde değiştirilecektir.

Bu işlemden sonra saklanmak istenen veri bitmişse eğer cari pikselin B değeri olduğu gibi atanacaktır. Saklanmak istenen veri bitene kadar bu işlem tekrarlanacaktır. Döngünün sonuna gelindiğinde ise anahtar saklanan resim pcbGonderilen kontrolünde gösterilecektir. Bu işlemden sonra Şifreli metni resmin esas tarafında saklamak olucaktır. Bu işlem Şekil 4.14'de gösterilmiştir.

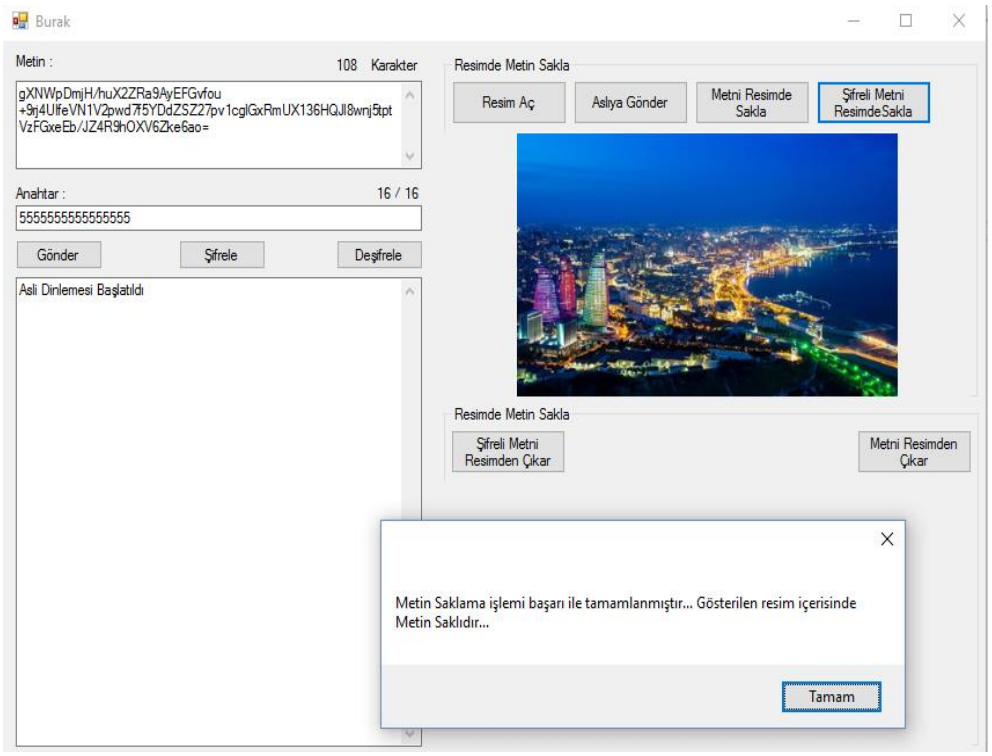
```
public void ResimdeSifreliMetinSakla(string Metin, Bitmap rsm)
{
    int R, G, B;
    int nR, nG, nB;

    int ResminPixelSayisi = rsm.Height * rsm.Width;

    string binaryMetin = StringToBinary(Metin);

    if (Metin.Length >= ResminPixelSayisi * 3)
    {
        MessageBox.Show("Girdiğiniz Metin Resime Sığdırılamıyor", "Uyarı");
    }
    for (int i = 1; i < rsm.Height; i++)
    {
        for (int j = 1; j < rsm.Width; j++)
        {
            Color piksel = rsm.GetPixel(j, i);

            R = piksel.R - piksel.R % 2;

            G = piksel.G - piksel.G % 2;

            B = piksel.B - piksel.B % 2;

            if (!string.IsNullOrEmpty(binaryMetin))
            {
                int SonKarbit1 = Convert.ToInt32(binaryMetin.Substring(binaryMetin.Length - 1, 1));

                nR = R + SonKarbit1;
                binaryMetin = binaryMetin.Remove(binaryMetin.Length - 1, 1);
                rsm.SetPixel(j, i, Color.FromArgb(nR, G, B));
                R = nR;
                if (string.IsNullOrEmpty(binaryMetin))
                {
                    rsm.SetPixel(j, i, Color.FromArgb(nR, piksel.G, piksel.B));
                    break;
                }
            }
            else
            {
                rsm.SetPixel(j, i, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
            }
            if (!string.IsNullOrEmpty(binaryMetin))
            {
                int SonKarbit2 = Convert.ToInt32(binaryMetin.Substring(binaryMetin.Length - 1, 1));
                nG = G + SonKarbit2;
                binaryMetin = binaryMetin.Remove(binaryMetin.Length - 1, 1);
                rsm.SetPixel(j, i, Color.FromArgb(R, nG, B));
                G = nG;
                if (string.IsNullOrEmpty(binaryMetin))
                {
                    rsm.SetPixel(j, i, Color.FromArgb(R, nG, piksel.B));
                    break;
                }
            }
            else
            {
                rsm.SetPixel(j, i, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
            }
            if (!string.IsNullOrEmpty(binaryMetin))
            {
                int SonKarBit3 = Convert.ToInt32(binaryMetin.Substring(binaryMetin.Length - 1, 1));
                nB = B + SonKarBit3;
                binaryMetin = binaryMetin.Remove(binaryMetin.Length - 1, 1);
                rsm.SetPixel(j, i, Color.FromArgb(R, G, nB));
                if (string.IsNullOrEmpty(binaryMetin))
                {
                    rsm.SetPixel(j, i, Color.FromArgb(R, G, nB));

                    break;
                }
            }
            else
            {
                rsm.SetPixel(j, i, Color.FromArgb(Convert.ToInt32(piksel.R), Convert.ToInt32(piksel.G), Convert.ToInt32(piksel.B)));
            }
        }
    }
    Random rnd = new Random();
    int resimAdi = rnd.Next();
    string CariProjeAdres = Path.GetDirectoryName(Path.GetDirectoryName(System.IO.Directory.GetCurrentDirectory()));
    pcbGonderilen.Image = rsm;
    rsm.Save(CariProjeAdres + "\\SteganoResimler\\" + "Stegano" + resimAdi.ToString() + ".png");

    MessageBox.Show("Metin Saklama işlemi başarı ile tamamlanmıştır... Gösterilen resim içerisinde Metin Saklıdır...");
}
```

Şekil 4.14: Şifreli Metni Resimde Saklama

Şekil 4.14'den de görüldüğü üzere Resimde Şifreli metin saklamak için öncelikle *int* veri tipinde resmin her piksel Kırmızı (Red), Yeşil (Green), Mavi (Blue) değerlerini tutmak için değişken tanımlanır. Bir sonraki aşamada veri saklandıkdan sonra bu renglerin deyişen değerlerini tutmak için ise *int nR, nG, nB* değişkenleri tanımlanır. Sığdırılmak istenen metnin Şifreli metin resmin tüm piksel sayısını aşarsa metin saklamak için bir başka resim seçilmesi gerekir. Bu işlemi kontrol etmel için her piksel 3 bitlik değer yerleştirilirse öncelikle resmin tüm piksel sayısı *int ResminPixelSayisi = rsm.Height*rsm.Width* şeklinde alınır. Resimde veri saklamak için, her saklanmak istenen her pikselin RGB değerlerinin her birinde en az öneme sahip olan bit yani RGB değerlerinin bit karşılığındaki veri "0" ile bitiyorsa bu son değerin yerine saklanması gereken verinin biti koyula bilir böylece resmin her pikselinin RGB değeri çok az farkla değiştirelecek fakat içerisinde saklanmak istenen veriyide saklayacaktır. Bu değişikliği insan göz farketmeyecektir. Bunun için saklanma istenen veri bit cinsine *string binaryMetin = StringToBinary(Convert.ToString(binaryMetin))* kodu ile çevrilerek string değere atanacaktır. Bir sonraki aşamada resmin tümündeki piksel sayısının 3 misli saklanmak istenen veriden büyük ise saklama işlemi gerçekleşecektir. Aksi taktirde " *girmek istediğiniz veri Resime sığdırlamıyor*" mesajı gösterilecektir. Metin saklama işleminde resmin piksellerindeki bütün bitler teker teker *Color piksel = rsm.GetPixel(j, i)* kod aracılığı ile ele alınır. Bir sonraki aşamada değişken olarak tanımlanan RGB değerlerinin her birinin sonu çift rakamla biten değerini atama nedeni ise sonu çift rakamla biten bir sayının bit karşılığı "0" ile bitmektedir ve bu değerin yerine saklanmak istenen verinin bit cinsinden son bitini koya biliriz. Bu işlemi yapmak için cari pikselin R,G,B değerlerinin *mod 2* göre hesaplıyoruz eğer bu işlem sonucu "1" ile biterse ki değerin tek rakamla bitmesi anlamına gelir. Bun nedenle cari pikselin, cari R,G,B değerlerine saklanmak istenen veriyi koyamayız bu yüzde cari R,G,B değerini bir azaltıyoruz ki bu değerler çift rakmla bitsin ve bu değerin bit karşılığındaki son son değeri olan "0" bitinin yerine verideki biti koya bilelim . Bir sonraki aşamada girilecek olan veri saklanması bitmişmi bitmemişmi kontrol ediyoruz ki bu kontrolün amacı bir sonraki döngüde saklanmak istenen veri bitmiş ola bilir. Döngünün ilk adımı olduğunu varsayarsak saklanmak istenen veri boş değildir. Bu veri boş olmadığına göre saklanmak istenen verinin son biti

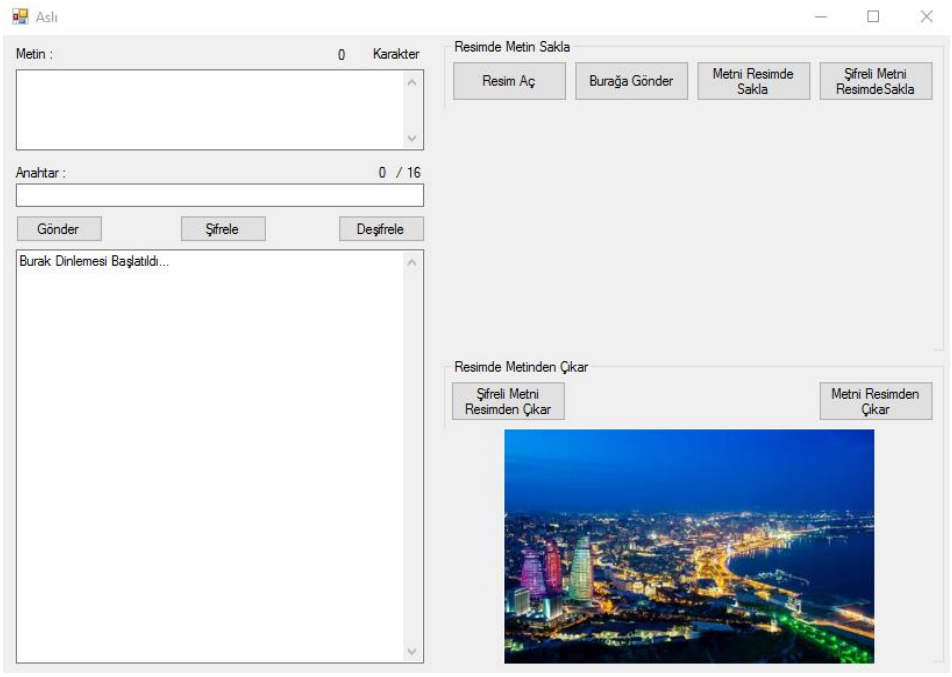
int SonKarbit1 = Convert.ToInt32(TextBoyutbit.Substring(binaryMetin.Length - 1, 1));

şeklinde alınır ve bu değer eğer "1" ise pikselin R değeri ile toplanarak yeni oluşacak bit yani nR değerinin bir üstü olacaktır, eğer bu değer "0" olucaksa yine toplama işlemi yapılarak değer olduğu gibi kalacaktır. Bit deyişmesi gerçekleştikten sonra o pikselin R *değerini rsm.SetPixel(j, i, Color.FromArgb(nR, G, B))* belirlemiş oluyoruz. Bu işlemden sonra saklanmak istenen veri bitmişse eğer cari pikselin G ve B değeri olduğu gibi bırakılacaktır ve döngünün sonuna gelinecektir. Saklanmak istenen veri bitmemişse eğer bir sonraki reng değeri olan G değeri aynı şekilde değiştirilecektir. Bu işlemden sonra saklanmak istenen veri bitmişse eğer cari pikselin B değeri olduğu gibi atanacaktır. Saklanmak istenen veri bitene kadar bu işlem tekrarlanacaktır. Döngünün sonuna gelindiğinde ise boyut saklanan resim pcbGonderilen kontrolünde gösterilecektir ve Şifreli metin, gizli anahtar Resimde saklı olacaktır bu işlem Şekil 4.15'de gösterilmiştir.



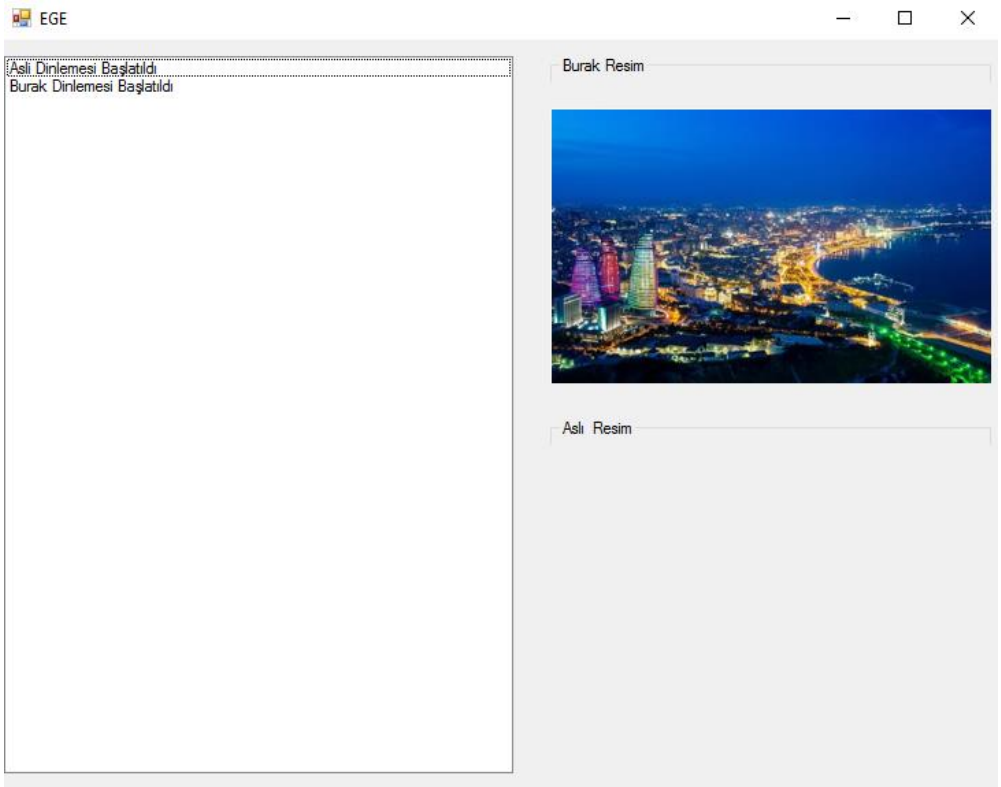
Şekil 4.15: Şifreli Metin Saklama İşlemi.

Burak Şifreli metni ve Gizli anahtarı resimde sakladıktan sonra "Aslıya Gönder " düymesine tıkladıktan sonra Resim Aslı uygulamasına gönderilecektir (Şekil 4.16).



Şekil 4.16: Aslıya gönderilen Resim.

Aslıya gönderilen resim aynı zamanda Pasiv Saldırgan Egeyede Gönderilmektedir (Şekil 4.17).



Şekil 4.17: Ege Uygulaması Arayüzü.

Burak Şifreli Metni Resimde sakladıktan sonra "Aslıya Gönder" düymesine tıkladığında çalışacak kod Şekil 4.17'de gösterilmiştir.

```
MemoryStream As_BellekAkim;
TcpClient Asli_Istemci;
NetworkStream Asli_R_AgAkimi;
BinaryWriter Asli_R_binaryYazici;

1 reference
public void Asli_ResimGonder()
{

    As_BellekAkim = new MemoryStream();


    Image rsm = pcbGonderilen.Image;
    rsm.Save(As_BellekAkim, System.Drawing.Imaging.ImageFormat.Png);

    byte[] on_bellek = As_BellekAkim.GetBuffer();
    As_BellekAkim.Close();

    Asli_Istemci = new TcpClient("localhost", 1992);


    Asli_R_AgAkimi = Asli_Istemci.GetStream();

    Asli_R_binaryYazici = new BinaryWriter(Asli_R_AgAkimi);

    Asli_R_binaryYazici.Write(on_bellek);

    Asli_R_binaryYazici.Close();

    Asli_R_AgAkimi.Close();

    Asli_Istemci.Close();
}
private void btnAslıyagonder_Click(object sender, EventArgs e)
{

    Asli_ResimGonder();
    Ege_ResimGonder();
}
```

Şekil 4.18: Aslıya Resim gönderen Kod.

Burak seçtiği resmi Aslıya göndermek için çalışan kod Şekil 4.17'den de görüldüğü gibi Resmi Socket üzerinden göndermek için resim *NetworkStream* tipine çevrilmeli bunun için *MemoryStream* klasının nesne örneği alınarak As_BellekAkim olarak belirlenir. Göndermek istenen resim *Image* tipindeki rsm değişkenine *Image rsm = pcbGonderilen.Image* kodu ile atamaktadır *rsm* değişkenini *MemoryStream* tipine dönüştürmek için *rsm.Save(As_BellekAkim, System.Drawing.Imaging.ImageFormat.Png);*

kodu kullanılmaktadır. Gönderilecek Resim *MemoryStream* tipine çevrildikten sonra resim ön belleğe *byte* tipinden diziye atanır. Bu işlem yapıldıktan Sonra As_BellekAkım kapatılır. Bu işlemden sonra Aslı uygulamasının dinlediği "1992" numaralı port numarasına bağlanılır. Gönderilecek Resim *NetworkStream* tipine çevrilmesi için *Asli_R_AgAkimi = Asli_Istemci.GetStream()* kodu kullanılır. Çevirme işleminden sonra BinaryWriter ile Resim karşı tarafa gönderilir. Bu işlemden sonra çalışan kodlar hepsi kapatılır. Burda yapılan işlemlerin aynısı Ege uygululaması için de yapılır ve Resim Hem Aslıya hemde Egeye gönderilmiş olucaktır. Gönderilmiş olan Resim Aslı tarafından kabül edilmesi için gereken kod Şekil 4.18'de gösterilmiştir.

```
public TcpListener rsmDinle;
public Socket rsmSocket;
public NetworkStream Rsm_AgAkim;

2 references
public void ResimDinle()
{
    try
    {
        rsmDinle = new TcpListener(1992);
        rsmDinle.Start();
        rsmSocket = rsmDinle.AcceptSocket();
        Rsm_AgAkim = new NetworkStream(rsmSocket);
        pcbGelen.Image = Image.FromStream(Rsm_AgAkim);
        rsmDinle.Stop();
        while(true)
        {
            ResimDinle();
        }
    }
    catch (Exception ex)
    {

        MessageBox.Show("Resim iletimindeki Sorun : "+ex.Message);
    }
}


  public Thread Paralel_Burak_Dinleyen;
  public Thread Paralel_Asli_Istemci;

  public Thread Paralel_Resim_Dinle;
  1 reference
  private void Form1_Load(object sender, EventArgs e)
  {

      Control.CheckForIllegalCrossThreadCalls = false;
      Paralel_Burak_Dinleyen = new Thread(Burak_Dinleyen);
      Paralel_Burak_Dinleyen.Start();

      Thread.Sleep(4000);
      Paralel_Asli_Istemci = new Thread(IstemciAsli);
      Paralel_Asli_Istemci.Start();

      Paralel_Resim_Dinle = new Thread(ResimDinle);
      Paralel_Resim_Dinle.Start();
  }
```

Şekil 4.18: Gönderilen Resmin alınması.

Burak tarafından gönderilecek resmin alınması için gereken kod Şekil 4.18'den de görüldüğü üzere Aslı Uygulamasının yüklendiği anda ResimDinle metodu çağrılmaktadır. ResimDinleMetodu hata ayıklama parantezleri içerisinde yazılma sebebi yaşanacak hatalarda projenin kapanmadan hata mesajını almaktır. *rsmDinle* kodu ile "1992" numaralı port dinlenilmektedir. Socketden gelecek Resmi Kabul etmek için çalışan kod *rsmSocket = rsmDinle.AcceptSocket()* kodudur. Socketdeki Resim *NetworkStream* tipinden olduğu için *NetworkStream* Tipinden olan *Rsm_AgAkim* değişkenine

atanmaktadır. Atama işleminden sonra Resmi almak için gereken kod *Rsm_AgAkim = new NetworkStream(rsmSocket)* şeklinde olacaktır. Bir sonraki adım ise gelen resmi pcbGelen kontrolünün *Image* özelliğine atanarak resim görselleştirilir.

Birsonraki adı ise Gelen Resimden Şifreli Mesaj ve Gizli Anahtarı çıkarmak için Aslı "Şifreli Metni Resimden çıkar" düymesine tıklayarak Şekil 4.19 da gösterilen kodları çalıştıracaktır.

```
public string ResimdenSifreliMetinBoyutCikar(Bitmap rsm)
{
    string SMetinBoyut = null;


    for (int i = 0; i < rsm.Height; i++)
    {
        Color piksel = rsm.GetPixel(0, i);
        if (piksel.R != 255 || piksel.G != 255 || piksel.B != 255)
        {
            if (piksel.R != 255)
            {
                string sonRbit = Convert.ToString(piksel.R % 2);
                SMetinBoyut += sonRbit;
            }
            if (piksel.G != 255)
            {

                string SonGbit = Convert.ToString(piksel.G % 2);
                SMetinBoyut += SonGbit;
            }

            if (piksel.B != 255)
            {
                string SonBbit = Convert.ToString(piksel.B % 2);
                SMetinBoyut += SonBbit;
            }
        }
    }

    SMetinBoyut = TersCevir(SMetinBoyut);

    SMetinBoyut = BinaryToString(SMetinBoyut);
    return SMetinBoyut;
}
```

Şekil 4.19: Şifreli Metnin Boyutunu Çıkarma

Şekil 4.19'dan da görüldüğü üzere ilk çalışacak kod Şifreli metnin boyutunu Resmin yüksekliğindeki ilk pikselleri okumak olacaktır. Okuma işlemi için bu metod parametre olarak *Bitmap* tipinden Resmi istemektedir. Geriye Şifreli Metnin Boyutunu göndermek için *string* tipinden *SmetinBoyut* değişkenine

başlangıç olarak “null” değerini atamaktadır. Bir sonraki adımda gelen resmin yüksekliğinde döngü ile dönerek her pikselin R,G,B değerlerini elde edilecektir. Pikseldeki R,G,B değerlerinden herhangi biri 255 den farklı ise şu demektir ki o pikselde gizletilmiş bit vardır, böylece koşulun içerisine girilerek bir sonraki koşulda pikselin R değeri 255 den farklı ise orda gizletilmiş bitin olduğu anlamına gelir ve o koşulun içerisine girilerek pikselin değerinin *mod 2* hesaplamasından çıkan sonuç “1” olursa gizletilmiş bit “1” demek olucaktır, eğer “0” olucaksa gizletilmiş bit “0” demektir. Bu işlemler pikselin diğer üyeleri olan G,B değerleri içinde tekrarlanıcaktır. Tekrarlanma sonucunda çıkıcak olan bit *TersCevir()* metodu ile tersine çevrilir ve *BinaryToString* metodu ile okunur hale Gelicektir sonra bu değer geriye Şifrelenmiş olan mesajın boyutudur ki geriye döndürülecektir. Bu işlem yapıldıktan sonra Şifrelenmiş Mesajın kendisi Şekil 4.20’de gösterildiği gibi metin kutusuna yazırılacaktır.

```
public string ResimdenSifreliMetinCikar(int SmtnBoyut, Bitmap rsm)
{
    string SMetin = null;
    SmtnBoyut = SmtnBoyut * 8;

    for (int i = 1; i < rsm.Height; i++)
    {
        for (int j = 1; j < rsm.Width; j++)
        {
            Color piksel = rsm.GetPixel(j, i);

            if (SmtnBoyut != 0)
            {
                string sonRbit = Convert.ToString(piksel.R % 2);
                SMetin += sonRbit;

                SmtnBoyut--;
            }
            if (SmtnBoyut != 0)
            {
                string sonGbit = Convert.ToString(piksel.G % 2);
                SMetin += sonGbit;
                SmtnBoyut--;
            }
            if (SmtnBoyut != 0)
            {
                string sonBbit = Convert.ToString(piksel.B % 2);
                SMetin += sonBbit;
                SmtnBoyut--;
            }
        }
    }
    SMetin = TersCevir(SMetin);

    SMetin = BinaryToString(SMetin);

    return SMetin;
}
```

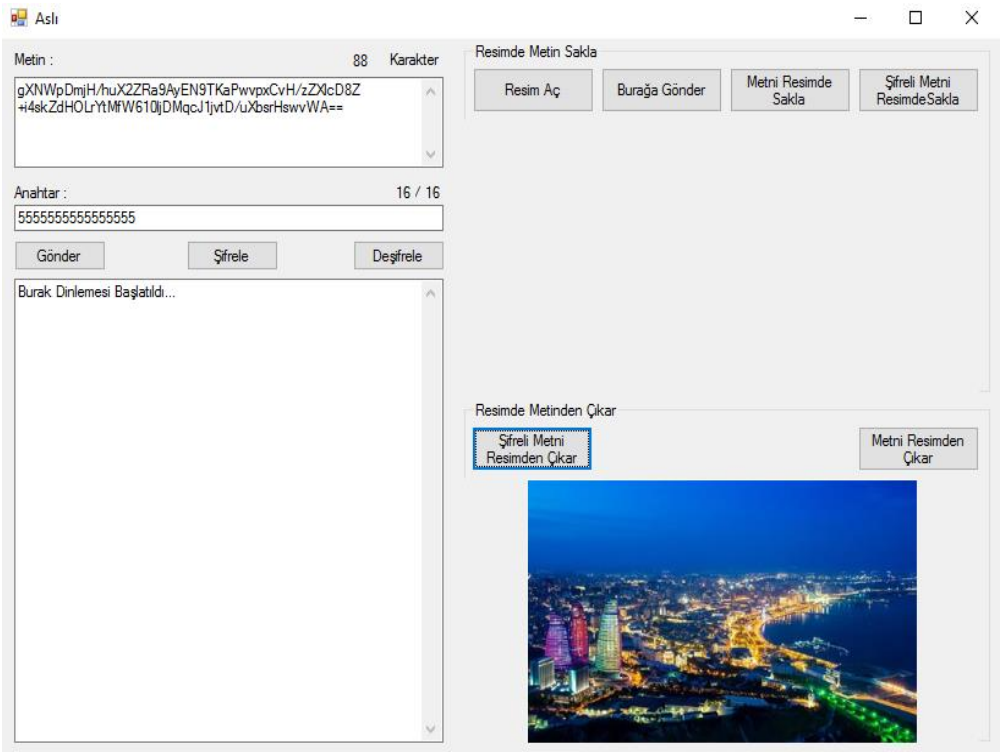
Şekil 4.20: Şifreli metnin Resimden çıkarma

Şekil 4.20'den de görüldüğü üzere ResimdenSifreliMetinCikar metodu parametre olarak *string* tipinden Sifreli Metnin Boyutunu ve Bitmap tipin resmi istemektedir.işlem sonucunda geriye SMetin gönderilecek ki başlangıç olarak "null" değeri atanmaktadır. Sifrelenmiş metnin boyutunu 8'e çarpma nedeni her karakterin 8 bittden olmasıdır. Bu işlemden sonra Resmin yükekliğindeki ve genişliğindeki piksellerin her birinin piksellerini *Color piksel = rsm.GetPixel(j, i)* kod aracılığı ile almaktadır. Bir sonraki aşamada metinBoyutunun "0" olmadığı takdirde koşulun parantezleri içerisine girecektir bu kontrol bir sonraki döngüde "0" eşit olduğunda Şifreli metnin tamamının çıktığı anlamına gelmektedir. Koşul sağlanıyor ise cari piksel R değeri *mod 2* 'e göre hesaplanır eğer burdan çıkan sonuç "1" ise Şifreli Metninin bit karşılığının değeri "1", eğer *mod* 2'e göder hesap sonucu "0" çıkacaksa verinin bit karşılığının değeri "0" olucaktır. Elde edilen bit değeri *SMetin* değişkeninin sonuna bu değer atanır. Atama işleminden sonra Şifreli metnin boyutu bir kere azaltılıcaktır. Eğer şifreli Metin boyutu "0" eşit olucaksa döngünün sonuna gelinecektir. Şifreli Metin Boyutu "0" olmadığı taktirde aynı işlem cari pikselin G değeri için yapılacaktır. Döngünün sonuna gelindiğinde ise *Smetin* değişkeni *TersCevir* metodu ile tersine çevrilir. Bir sonraki aşamada *SMetin* içerisinde bulunan bitler *BinaryToString* metodu ile okunur hale gelip geriye döndürülecektir. Geriye döndürülen değer metin kutusunda Şifreli metin olarak görselleşicektir. Bir sonraki aşamada Şifreli metni deşifrelemek için gereken gizli anahtarı Resim içerisinde çıkartma işlemi olucaktır ki bu işlem Şekil 4.21' de gösterilmiştir.

```
public string ResimdenSifreCikar(Bitmap rsm)
{
    string Sifre = null;

    int SifreBoyut = 128;
    for (int j = 1; j < rsm.Width; j++)
    {
        Color piksel = rsm.GetPixel(j, 0);
        if (SifreBoyut!=0)
        {
            string SonSbit = Convert.ToString(piksel.R % 2);
            Sifre += SonSbit;
            SifreBoyut--;
        }
        if (SifreBoyut!=0)
        {
            string SonSbit2 = Convert.ToString(piksel.G % 2);
            Sifre += SonSbit2;
            SifreBoyut--;
        }
        if (SifreBoyut!=0)
        {
            string SonSbit3 = Convert.ToString(piksel.B % 2);
            Sifre += SonSbit3;
            SifreBoyut--;
        }
    }
    Sifre = TersCevir(Sifre);
    Sifre = BinaryToString(Sifre);


    return Sifre;
}
```

Şekil 4.21: Gizli anahtarın Resimden çıkarılması

Şekil 4.21'den de görüldüğü üzere *ResimdenSifreCikar* Metodu parametre olarak *Bitmap* tipinden resim almaktadır. Metodun içerisinde geri döndürülecek olan *string* tipinden *Sifre* başlangıç olarak "null" değeri atanmıştır. AES algoritmasının gizli anahtar boyutu 128 bit olduğundan şifre çıkarma işlemi döngü içerisinde 128 defa tekrarlanıcaktır. Bu döngü ile resmin genişliğinde hareket edilerek *Color piksel = rsm.GetPixel(j, 0)* kodu ile tüm pikseller teker teker ele alınır. Bir sonraki aşamada *SifreBoyutunun* "0" olmadığı taktirde koşulun parantezleri içerisine girilir. Burada cari pikselin R değeri *mod 2*' e göre hesaplanır burdan çıkan "0" ve ya "1" gizli anahtarın bit karşılığının değeri olucaktır. Bit karşılı *Sifre* değişkeninin sonuna eklenecektir. Bir sonraki aşamada SifreBoyut 1 kere azaltılacaktır. Bir sonraki koşul gecerli olursa aynı işlem G değeri için yapılacaktır. Koşul geçerli olmadığı taktirde ise döngünün sonuna gelinerek *Sifre* değişkenindeki veri *TersCevir* metoduyla tersine çevrilir ve sonra bu bitlar *BinaryToString* metodu ile okunur hale gelerek geri döndürülür. Geri döndürülen değer arayüzdeki anahtar kutusunda görselleşicektir ki bu Şekil 4.22 de gösterilmiştir.



Şekil 4.22: Şifreli metin ve Gizli anahtarın Resimden Çıkarılması

Böylece gizli anahtar ve Şifreli mesaj güvensiz kanal üzerinden başarıyla gönderilmiştir.

5 SONUÇ

Bu tezim amacı Haberleşme esnasın olabilecek pasiv saldırılara karşı dirençliliğin gösterilmesi için yapılan önlemlerin alınmasıdır. Tez kapsamında yapılan projede temel amaç, gizli anahtarlı şifreleme yöntemi iletişimde olan taraflar iletişim esnasında yapılmış ortadaki adam saldırısını gerçekleştiren saldırganın, iletişimin şifreli olduğuna dair şüphelerinin olmaması, ve aynı zamanda gizli anahtarlı şifrelemede şifreleme ve deşifreleme işlemlerini gerçekleştirecek anahtarı güvenli kanal olmadığı taktirde karşı tarafa gizli anahtarı Steganografi tekniği ile iletmektir. Bu iletişimde mesajı gönderen taraf, mesasjı belirlediği 16 karakterli anahtar ile şifreler ve karşı tarafa bu şifreli mesajı ve gizli anahtarı "png" formatlı resmin içerisinde saklayarak karşı tarafa göndermelidir. Saldırgan bu resmide elde eder fakat bu resmin içerisinde birşeylerin saklandığını farkedemeyecektir.

Tez kapsamında yapılan proje Visual Studio platformunun C# programlama dilinde yazılmıştır. Bu projede yapay olarak ortadaki adam pasiv saldırısı yapılmaktadır. Projede yapılan teknikler vasıtası ile Socketler üzerinden özel haberleşme güvenli olarak gerçekleştirile bilinir.

KAYNAKLAR

[1] **Niels Ferguson, Bruce Schneier, Tadayoshi Kohno.** *"Cryptography Engineering".* Published by Wiley Publishing, Inc in 2010.

[2] **Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone.** *"HANDBOOK of APPLIED CRYPTOGRAPHY".* Massachusetts Institute of Technology June 1996

[3] **ALAN G. KONHEIM.** *"COMPUTER SECURITY AND CRYPTOGRAPHY".* Published by John Wiley & Sons in 1934.

[4] **Mark Stamp, Richard M. Low.** "*APPLIED CRYPTANALYSIS*". Published by JOHN WILEY & SONS, INC in 1996

[5] **Christof Paar, Jan Pelzl.** *"Understanding Cryptography"*. Published by Springer in 2010

[6] **MILES E. SMID, DENNIS K. BRANSTAD.** *"Contemporary Cryptology".* Published by National Institute of Standards and Technology in 1980.

[7] **Laurens Van Houtven.** "*Crypto101*". Published by Creative Commons Attribution in 2013

[8] **Celine Blondeau, Gregor Leander, Kaisa Nyberg.** *"Differential-Linear Cryptanalysis Revisited".* Published by Faculty of Electrical Engineering and Information Technology, Ruhr Universita¨t Bochum, Germany in 2013

[9] **Daemen, J. ve Rijmen, V.,** 2002. "*The Design of Rijhndael AES-The Advanced Encryption Standart*".

[10] **Yuliang Zheng, Josef Pieprzyk, Jennifer Seberry.** *"A one-way hashing algorithm with variable length output".* Published by University of Wollongong in 1993

[11] **Nigel Smart.** *"Cryptography Smart".* Published in 2003

[12] **Edward Schaefer.** *"An introduction to cryptography and cryptanalysis"*. Santa Clara University in 1998

[13] **RSA Laboratories**. Answers to frequently asked questions about today's cryptography, version 3.0. RSA Data Security, Inc., 1996.

[14] **Mitsuru Matsui.** *"Linear Cryptanalysis Method for DES"*. Computer and Information System Laboratory Mitsubishi Electrical Corparation. 1998

[15] **R. Rivest.** *"The MD5 Message-Digest Algorithm"* MIT Laboratory for Computer Science and RSA Data Security, Inc. April 1992

[16] **Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, San Ling.** *"Improved Meet-in-the-Middle Cryptanalysis of KTANTAN".* Published by Division of Mathematical Sciences, School of Physical and Mathematical Sciences Nanyang Technological University, Singapore in 1980

[17] **Orr Dunkelman.** "*Hash Functions — MD5 and SHA1*". Published by University of Haifa in 14 March 2012.

[18] **C. E. SHANNON.** *"Communication Theory of Secrecy Systems".* Published by Bell System Technical Journal in July 1948

[19] **Stinson, D.R.** *"Cryptography, Chapman & Hall".* Published by CRC Press Company in 2002

[20] **Bart Preneel.** *"CRYPTANALYSIS AND DESIGN OF SYNCHRONOUS STREAM CIPHERS".* Published by KATHOLIEKE UNIVERSITEIT LEUVEN in June 2006

[21] **Savaş, E.** *". Dizi Şifreleme Sistemleri ve Doğrusal Karmaşıklık"***.** Yüksek Lisans Tezi, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.19

ÖZGEÇMİŞ

Ad- Soyad : Orkhan ABDULLAZADA

Doğum Tarihi ve Yeri : 1992.02.02 Azerbaycan

E-Posta : orhanyazilimci@gmail.com

ÖĞRENİM DURUMU

- **Lisans** : 2013, Batı Üniversitesi, Bilgisayar Mühendisliği
- **Yüksek Lisans** : İstanbul Aydın Üniversitesi, Bilgisayar Mühendisliği