

# CLIENT-SERVER ŞİFRELEME UYGULAMASI

Numara: 436550

İsim: Emre Üçbudak

Github Repo: <https://github.com/emreucbudak/KriptolojiOdev>

## 1. Projenin Amacı ve Kapsamı

Bu projenin temel amacı, Kriptoloji dersi kapsamında teorik olarak öğrenilen şifreleme yöntemlerinin çalışma mantığını kavramak, algoritmaların iç yapılarını (round, permütasyon, bit işlemleri) deneyimlemek ve bunları gerçek bir yazılım mimarisi üzerinde uygulamaktır.

Proje, **C#** programlama dili kullanılarak **TCP/IP** protokolü üzerinde çalışan, çoklu iş parçacığı (multithreading) destekli bir **İstemci-Sunucu (Client-Server)** uygulaması olarak geliştirilmiştir. Uygulama kapsamında; Sezar, Vigenere, Yer Değiştirme gibi klasik şifreleme tekniklerinin yanı sıra, **AES** ve **DES** gibi simetrik blok şifreleme standartları ve **RSA** asimetrik şifreleme algoritması kodlanmıştır.

Ödev gereksinimleri doğrultusunda sistem **iki farklı modda** çalışmaktadır:

- Kütüphane Tabanlı Mod:** .NET kriptografi kütüphaneleri kullanılarak AES, DES ve RSA ile güvenli veri iletimi sağlanmıştır.
- Manuel (Kütüphanesiz) Mod:** Blok şifreleme mantığının pekiştirilmesi amacıyla, **DES algoritması** herhangi bir hazır kütüphane kullanılmadan, bit seviyesinde işlemler ve Feistel ağı yapısı kullanılarak **manuel olarak** implemente edilmiştir.

Bu çalışma sayesinde, açık metinlerin (plaintext) şifreli metne (ciphertext) dönüştürülmesi ve ağ üzerinden iletilmesi süreçleri simüle edilerek veri gizliliği (confidentiality) sağlanmıştır. Ayrıca, **Wireshark** ağ analiz aracı kullanılarak iletilen TCP paketleri yakalanmış; şifreli verinin ağ üzerindeki görünümü ve AES/RSA algoritmalarının paket boyutlarına etkisi uygulamalı olarak analiz edilmiştir.

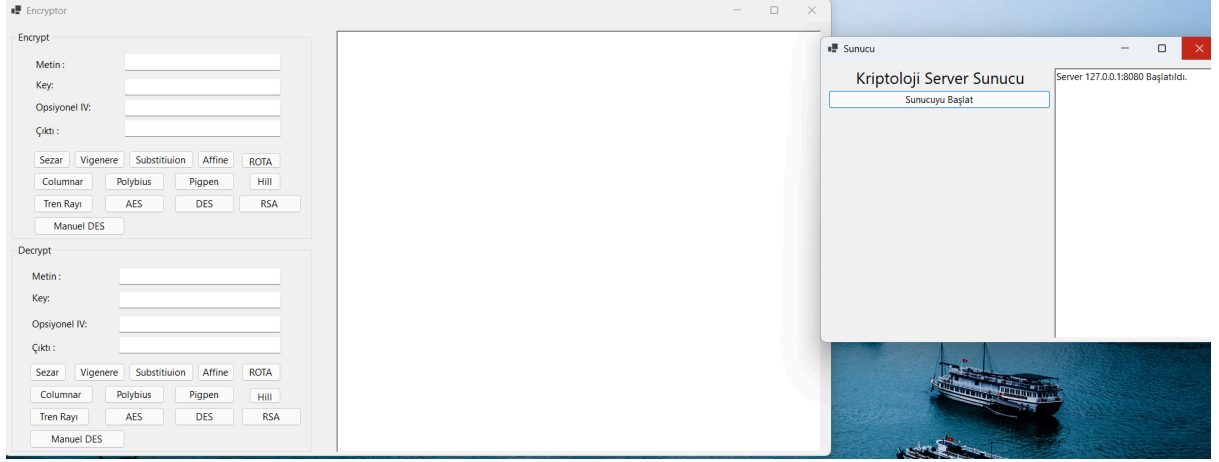
## 2. Sistem Mimarisi ve Çalışma Akışı

Uygulama, dağıtık bir yapı olan İstemci-Sunucu (Client-Server) mimarisi üzerine kurulmuştur. İletişim protokolü olarak veri bütünlüğünü garanti eden **TCP (Transmission Control Protocol)** tercih edilmiştir.

### 2.1. Genel Akış

- Sunucu (Server):** Belirlenen IP (127.0.0.1) ve Port (8080) üzerinden dinleme (Listening) moduna geçer. Çoklu istemci desteği için her gelen bağlantıyı ayrı bir **Thread** üzerinde karşılar.
- İstemci (Client):** Kullanıcı arayüzü üzerinden metin, anahtar (Key) ve (opsiyonel) IV (Initialization Vector) bilgilerini alır.
- Şifreleme ve Paketleme:** Seçilen algoritmaya göre veri şifrelenir. Şifreli veri, işlem türü ve algoritma adıyla birleştirilerek (**Op|Algo|Text|Key|IV** formatında) sunucuya gönderilir.
- Sunucu İşleme:** Sunucu gelen paketi ayrıştırır (Parsing), ilgili şifreleme/çözme servisini çağırır ve sonucu istemciye geri döner (Echo/Response).

Şekil 1: İstemci ve Sunucu Uygulamasının Genel Görünümü



### 3. Kullanılan Algoritmalar ve Karşılaştırma

Projede üç temel modern algoritma kullanılmıştır: AES, DES ve RSA.

#### 3.1. AES (Advanced Encryption Standard)

Simetrik bir blok şifreleme algoritmasıdır. Projede 128-bit blok boyutu ve 256-bit anahtar desteği ile kullanılmıştır. Güvenliği ve hızı nedeniyle günümüzün standart algoritmasıdır.

#### 3.2. DES (Data Encryption Standard)

Eski bir simetrik standarttır. 64-bit blok boyutu ve 56-bit anahtar kullanır. Projede hem **.NET** kütüphanesi ile hem de Manuel olarak kodlanmıştır. Günümüzde güvensiz kabul edilse de Feistel yapısını öğrenmek için idealdir.

#### 3.3. RSA (Rivest–Shamir–Adleman)

Asimetrik bir şifreleme algoritmasıdır. Bir adet Public Key (Şifreleme için) ve bir adet Private Key (Çözme için) kullanır. Projede 2048-bit anahtar uzunluğu tercih edilmiştir. Simetrik şifrelemeye göre çok daha yavaştır, bu nedenle genellikle büyük verileri şifrelemek yerine simetrik anahtarların güvenli dağıtımı (Key Exchange) için kullanılır.

Uygulama senaryosunda; kullanıcı RSA şifreleme butonuna tıkladığında, istemci tarafında **.NET** kütüphaneleri kullanılarak anlık (ephemeral) bir anahtar çifti üretilir. Açık anahtar (Public Key) ile metin şifrelenir ve şifreli veri sunucuya iletilir.

Özellik	AES	DES	RSA

Tür	Simetrik	Simetrik	Asimetrik
Anahtar Boyutu	128, 192, 256 bit	56 bit	1024, 2048+ bit
Blok Boyutu	128 bit	64 bit	Anahtar boyutuna bağlı
Hız	Çok Hızlı	Hızlı	Yavaş
Kullanım Alanı	Veri Şifreleme	Eğitim/Legacy	Anahtar Dağıtımı / İmza

## 4. Manuel ve Kütüphane Tabanlı Şifreleme Analizi

Ödevin en kritik aşamalarından biri olan "Kütüphanesiz Şifreleme" için DES algoritması seçilmiştir. `EncryptorService` sınıfı içerisinde `ManuelDesEncrypt` metodu geliştirilmiştir.

### 4.1. Manuel İmplementasyon Detayları

Hazır fonksiyonlar (`DESCryptoServiceProvider` vb.) kullanılmadan, algoritmanın temel yapı taşları kodlanmıştır:

- Veri 64-bitlik (8 byte) bloklara bölünmüştür.
- Bitwise (Bit seviyesi) operatörler (`<<`, `>>`, `^`, `&`) kullanılarak veriler üzerinde manipülasyon yapılmıştır.
- Basitleştirilmiş bir Feistel Ağı yapısı kurularak, verinin sağ ve sol yarıları (L ve R) 16 döngü boyunca anahtarla işleme sokulmuştur.

### 4.2. Karşılaştırma Sonucu

- Performans: Kütüphane tabanlı DES, optimize edilmiş C++ altyapısını kullandığı için manuel koddan daha hızlı çalışmaktadır.
- Teknik Analiz: Standart kütüphanelerin arka planda otomatik olarak yönettiği veri dolgulama (Padding) ve bloklar arası bağımlılık (CBC Modu) süreçleri, manuel kodlama sayesinde bit seviyesinde analiz edilerek deneyimlenmiştir.

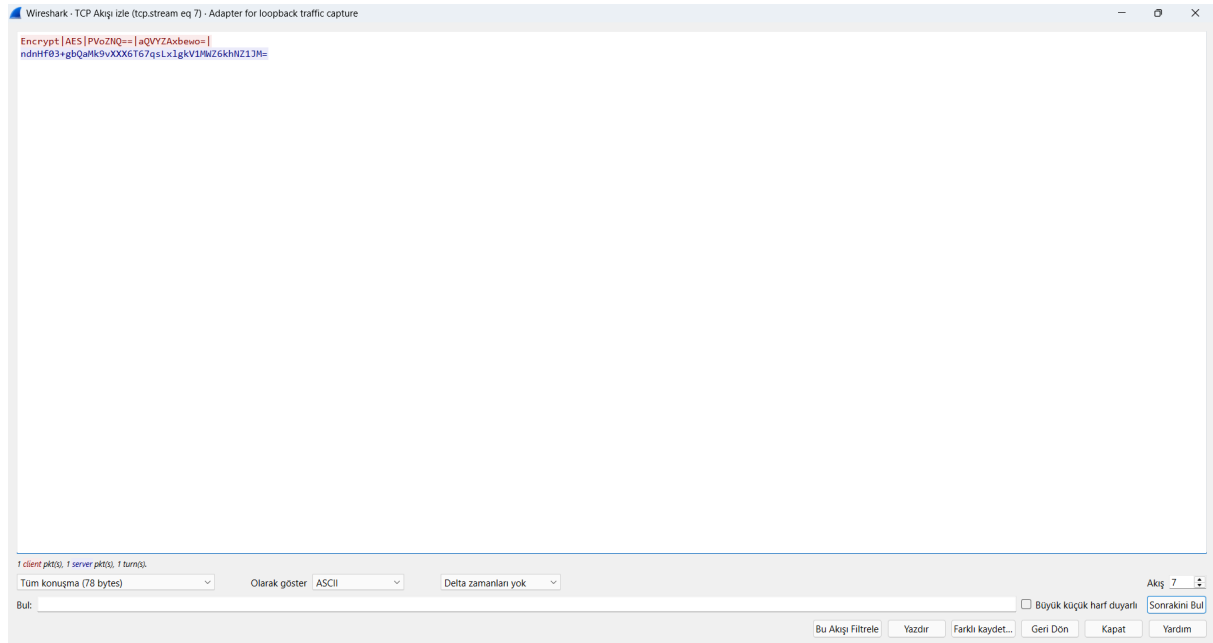
## 5. Wireshark ile Ağ Trafiği Analizi

Uygulamanın TCP üzerinden gerçekleştirdiği veri transferi Wireshark aracı ile dinlenmiş ve analiz edilmiştir.

### 5.1. Şifreli Verinin Görünümü

Aşağıdaki ekran görüntüsünde, istemciden sunucuya gönderilen bir paketin içeriği görülmektedir. Veri AES ile şifrelendiği için "Payload" kısmı tamamen anlamsız karakterlerden oluşmaktadır. Bu durum, ağdaki bir saldırganın (Man-in-the-Middle) veriyi yakalasa bile içeriğini okuyamayacağını kanıtlar.

Şekil 2: AES ile Şifrelenmiş TCP Paket İçeriği (Okunamaz Veri)

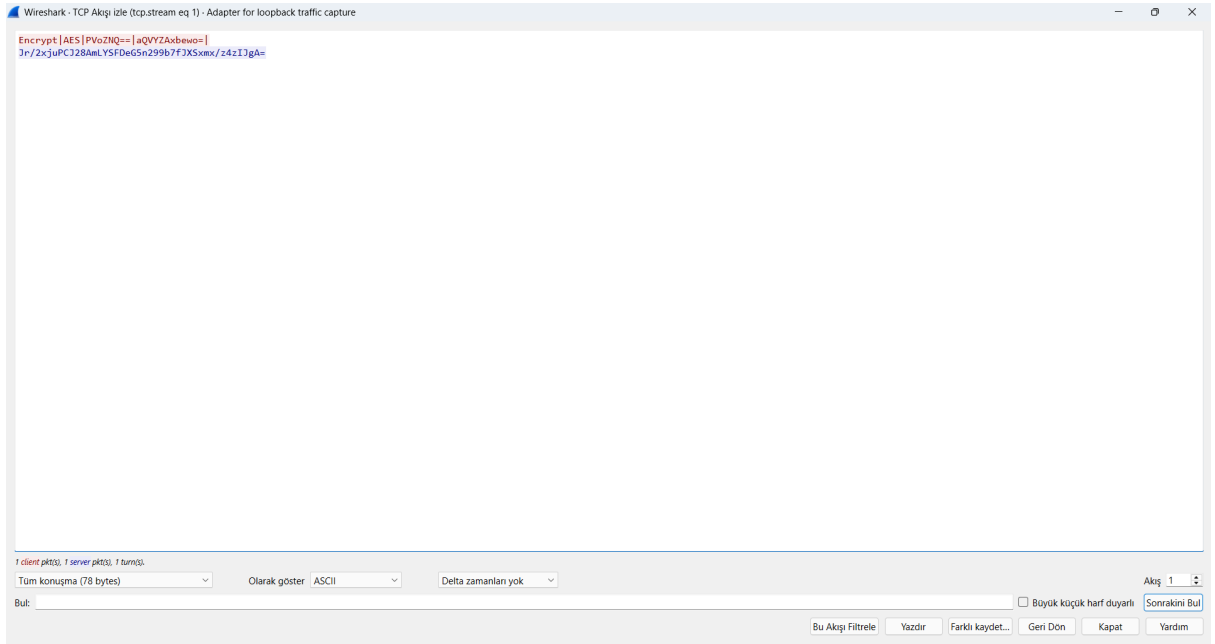
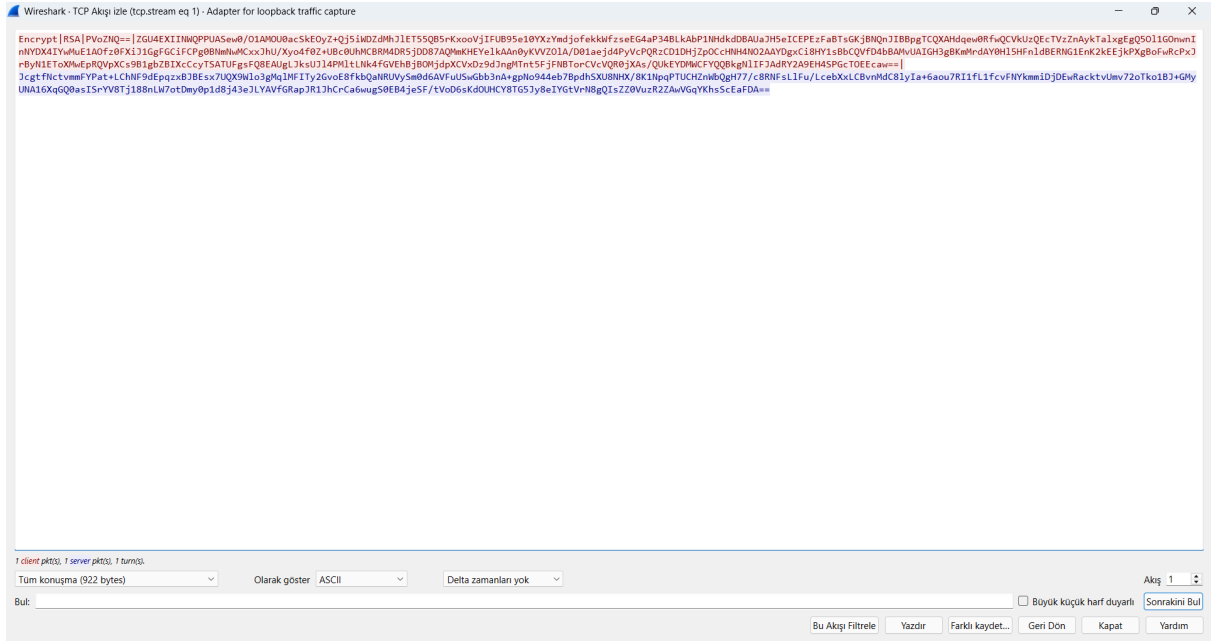


### 5.2. Paket Boyutlarının Karşılaştırması (AES/DES vs RSA)

Analiz sırasında en dikkat çekici fark, paket boyutlarında gözlemlenmiştir.

- **AES/DES:** Blok boyutları küçük (128/64 bit) olduğu için, kısa metinlerde paket boyutu minimum seviyededir.
- **RSA:** 2048-bit anahtar kullanıldığında, şifreli çıktı (Ciphertext) doğrudan anahtarın modül boyutuna (256 byte) denk gelmektedir. Bu nedenle çok kısa bir metin ("Merhaba") bile şifrelense, RSA paket boyutu AES paket boyutuna göre oldukça büyüktür.

Şekil 3: RSA Algoritmasının Paket Boyutuna Etkisi



Bu analiz, RSA'nın neden büyük verileri şifrelemek için değil, sadece küçük boyutlu AES anahtarlarını şifrelemek (Key Exchange) için kullanıldığını teknik olarak doğrulamaktadır.

## 6. Sonuç

Bu proje ile simetrik (AES, DES) ve asimetrik (RSA) şifreleme algoritmalarının çalışma prensipleri hem teorik hem de pratik olarak incelenmiştir. Özellikle:

1. Manuel DES implementasyonu ile blok şifrelemenin iç yapısı (Bit manipölasyonu, Feistel yapısı) kodlanarak öğrenilmiştir.

2. TCP Soket Programlama ile güvenli bir haberleşme kanalı oluşturulmuştur.
3. Wireshark Analizi ile şifrelemenin ağ trafiği üzerindeki etkisi (gizlilik ve paket boyutu artışı) somut verilerle gözlemlenmiştir.

Proje, modern kriptografik sistemlerin temel yapı taşlarını anlamak adına başarılı bir simülasyon olmuştur.