CLİENT-SERVER ŞİFRELEME UYGULAMASI

1. Projenin Amacı ve Kapsamı

Bu projenin temel amacı, **Kriptoloji** dersi kapsamında teorik olarak öğrenilen şifreleme yöntemlerinin çalışma mantığını kavramak ve bunları gerçek bir yazılım mimarisi üzerinde uygulamaktır.

Proje, **C#** programlama dili kullanılarak **TCP/IP** protokolü üzerinde çalışan bir İstemci-Sunucu (Client-Server) uygulaması olarak geliştirilmiştir. Uygulama kapsamında; **Sezar, Vigenere, Yer Değiştirme** gibi klasik şifreleme teknikleri ile **AES ve DES** gibi modern blok şifreleme standartları kodlanmıştır.

Bu çalışma sayesinde, açık metinlerin (plaintext) seçilen algoritma ve anahtar ile şifreli metne (ciphertext) dönüştürülmesi, ağ üzerinden güvenli bir şekilde iletilmesi ve alıcı tarafta tekrar çözülmesi süreçleri simüle edilerek **veri gizliliğinin (confidentiality)** sağlanması hedeflenmiştir.

---

2. Kullanılan Teknolojiler ve Yöntemler

Projenin geliştirilme sürecinde, modern yazılım geliştirme standartlarına uygun, güvenli ve ölçeklenebilir bir yapı oluşturmak adına aşağıdaki teknolojiler ve kütüphaneler kullanılmıştır:

- **Programlama Dili ve Platform:** Proje, güçlü tip güvenliği ve geniş kütüphane desteği nedeniyle **C# (C Sharp)** programlama dili kullanılarak, **.NET Framework** altyapısı üzerinde geliştirilmiştir.
- **Kullanıcı Arayüzü (UI):** Kullanıcı etkileşimini sağlamak ve şifreleme süreçlerini görselleştirmek amacıyla **Windows Forms (WinForms)** teknolojisi kullanılmıştır.
- **Ağ İletişimi (Networking):** İstemci (Client) ve Sunucu (Server) arasındaki haberleşme, veri bütünlüğünü garanti eden **TCP/IP (Transmission Control Protocol)** protokolü ve **Socket Programlama** mimarisi ile sağlanmıştır.
- **Asenkron Programlama:** Ağ işlemleri sırasında arayüzün donmasını engellemek ve performansı artırmak için **Task Asynchronous Pattern (Async/Await)** yapısı kullanılmıştır.

---

3. Yazılım Mimarisi ve Kod Yapısı

Proje, sürdürülebilirlik, genişletilebilirlik ve performans kriterleri göz önüne alınarak Modüler Katmanlı Mimari (Modular Layered Architecture) prensiplerine göre tasarlanmıştır. Uygulamanın kod yapısı, Kullanıcı Arayüzü (UI), İş Mantığı (Business Logic) ve Ağ İletişimi (Networking) olmak üzere üç temel mantıksal katmana ayrılmıştır.

3.1. İstemci-Sunucu (Client-Server) Mimarisi

Uygulama, dağıtık sistemlerin temeli olan Client-Server topolojisi üzerine kurulmuştur.

- **İstemci (Client): Kullanıcıdan veriyi (metin, anahtar, algoritma seçimi) toplar ve bir Protokol Paketi haline getirerek sunucuya iletir. Şifreleme işlemi istemci tarafında değil, merkezi bir otoriteyi simüle eden sunucu tarafında gerçekleştirilir.**
- **Sunucu (Server): TCP portunu (varsayılan: 8001) dinler, gelen istekleri asenkron olarak kabul eder, ilgili kriptografik işlemi uygular ve sonucu istemciye geri döner.**

3.2. Soyutlama ve Arayüz (Interface) Kullanımı

Projede SOLID prensiplerine sadık kalmak ve kod tekrarını önlemek amacıyla Arayüz Tabanlı Programlama (Interface-Based Programming) kullanılmıştır.

- **`IEncryptorService` ve `IDecryptorService`: Tüm şifreleme ve çözme algoritmaları bu arayüzler üzerinden tanımlanmıştır. Bu sayede, gelecekte projeye yeni bir algoritma (örneğin RSA) eklenmek istendiğinde, mevcut kod yapısını bozmadan sadece ilgili metoda ekleme yapmak yeterlidir.**
- **Bu yapı, kriptografik işlemlerin Loose Coupling (Gevşek Bağlılık) prensibiyle ana uygulamadan soyutlanmasını sağlamıştır.**

3.3. Servis Odaklı Kod Yapısı (Service Classes)

Kriptografik algoritmalar form (UI) kodlarının içine gömülmemiş, ayrı sınıflar (Class) halinde izole edilmiştir:

- **`EncryptorService.cs`: AES (Pure), DES (System) ve Klasik şifreleme algoritmalarının mantıksal kodlarını barındırır.**
- **`DecryptorService.cs`: Şifreli metinlerin çözülmesi için gerekli ters fonksiyonları içerir.**
- **Bu ayrım, kodun okunabilirliğini artırmış ve hata ayıklama (debugging) süreçlerini kolaylaştırmıştır.**

3.4. Özel Uygulama Protokolü (Application Layer Protocol)

TCP üzerinden giden verinin anlamlı hale gelmesi için metin tabanlı özel bir protokol tasarlanmıştır. İletişim şu formatta gerçekleşir: `İşlemTürü | Algoritma | Tünellenmiş Veri (XOR+Base64) | Anahtar | IV`

- **Paket Yapısı: `Encrypt|DES|aGVsbG8=|123456|...`**

- **Sunucu bu paketi `|` (pipe) karakterine göre ayrıştırarak (parsing), işlem türünü ve parametreleri belirler.**

3.5. İletim Katmanı Güvenliği ve Veri Tünelleme (Transport Security)

Ağ üzerindeki paket analizi (sniffing) saldırılarına karşı, istemci ve sunucu arasında XOR Cipher tabanlı bir Güvenli Tünel (Secure Tunnel) mekanizması kurulmuştur.

- **İşleyiş: Kullanıcının girdiği açık metin (plaintext), ağa gönderilmeden önce istemci tarafında XOR algoritması ile şifrelenir ve Base64 formatına çevrilir.**
- **Sunucu Tarafı: Sunucu, gelen paketi aldığında önce bu XOR katmanını (tüneli) çözer ve orijinal metne ulaşır. Ardından kullanıcının talep ettiği asıl şifreleme algoritmasını (AES, DES, Sezar vb.) bu temiz metin üzerinde uygular.**
- **Bu sayede, ağ trafiği dinlense dahi (Wireshark vb.) saldırganlar ham metni veya anahtarı göremez, sadece anlamsız veri yığınlarıyla karşılaşır.**

3.6. Asenkron Programlama (Async/Await)

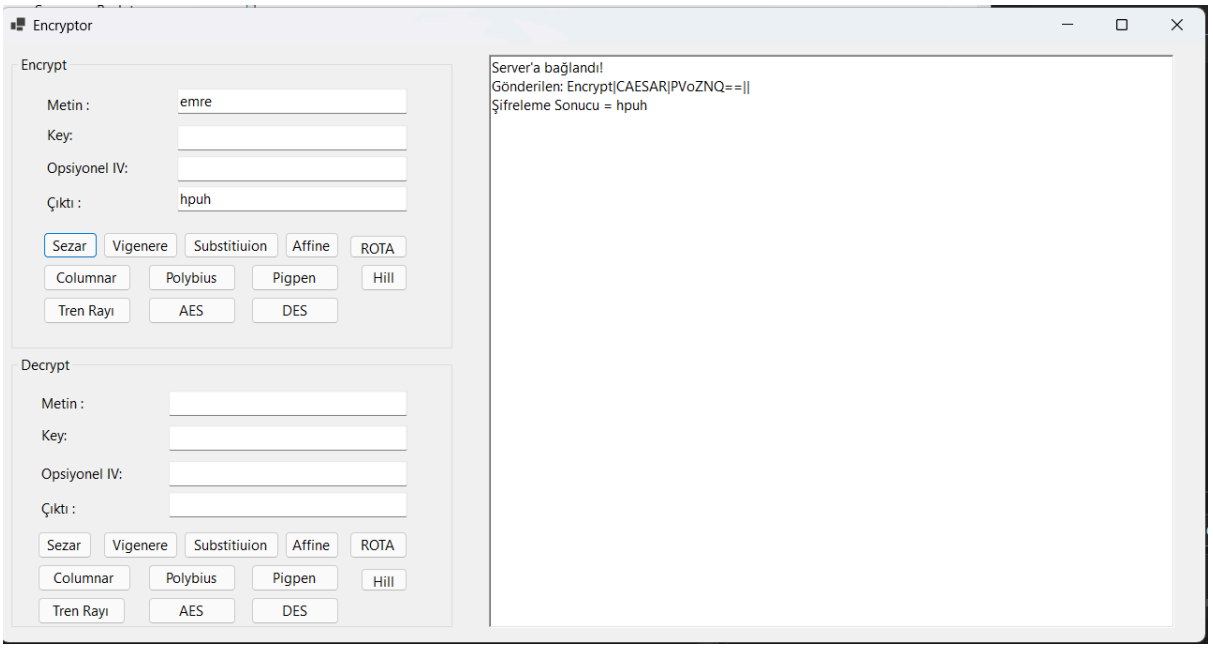
Ağ trafiği sırasında veya yoğun matematiksel işlem gerektiren AES/DES döngüleri sırasında kullanıcı arayüzünün donmasını (UI Freezing) engellemek için .NET Task Asynchronous Pattern (TAP) kullanılmıştır.

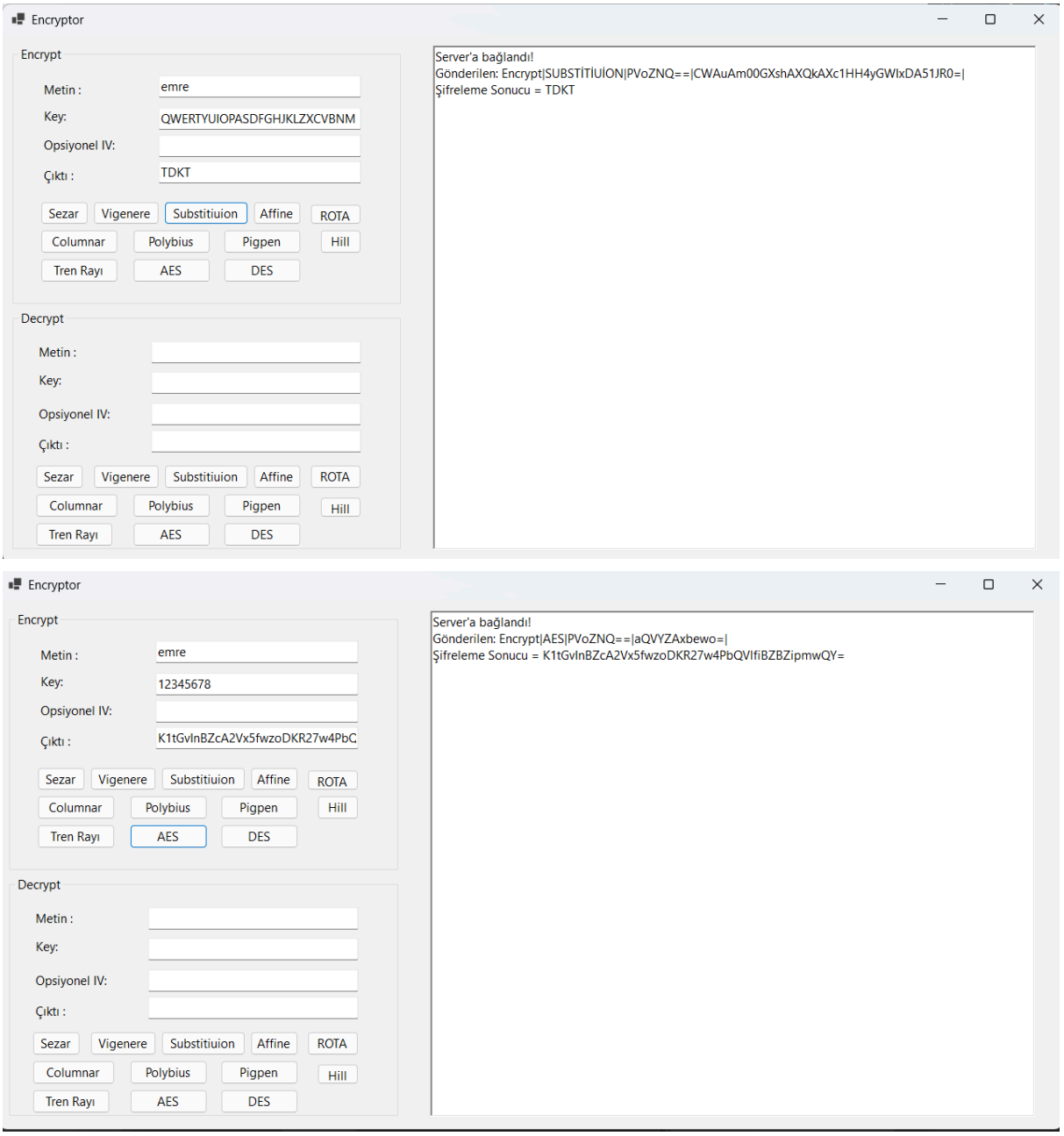
- **`ConnectToServer`, `GetStream` ve şifreleme operasyonları `await` anahtar kelimesi ile çağrılarak, ana işlem parçacığının (Main Thread) bloklanması engellenmiş, akıcı bir kullanıcı deneyimi sağlanmıştır.**

---

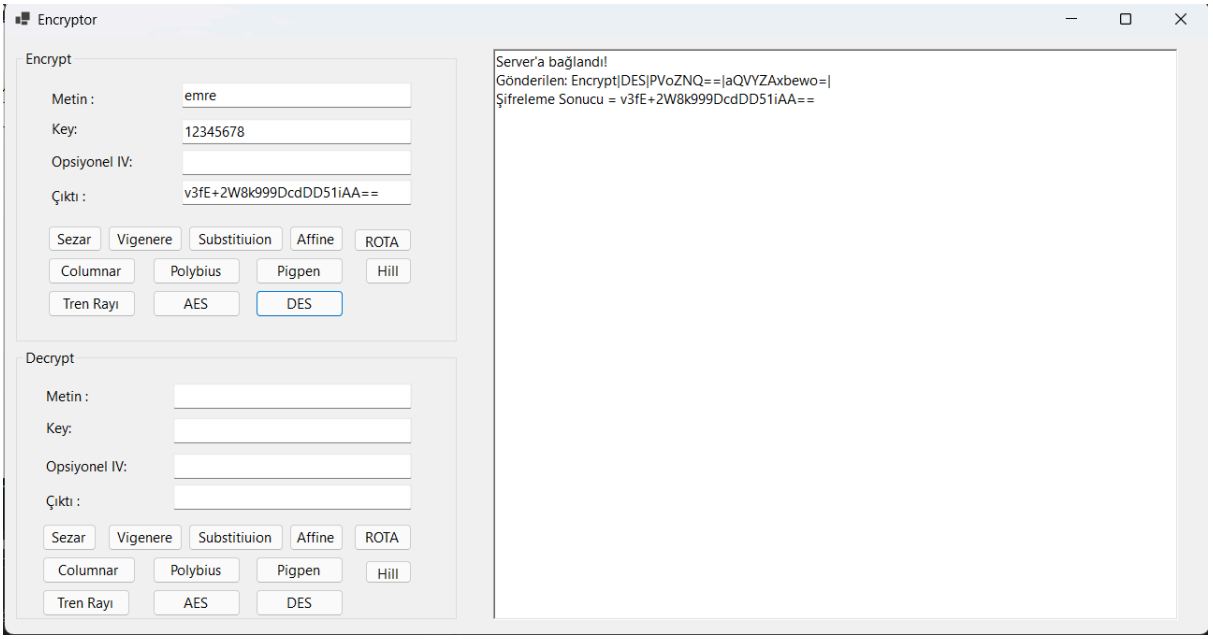
4. Projenin GUI ve Wireshark örnekleri

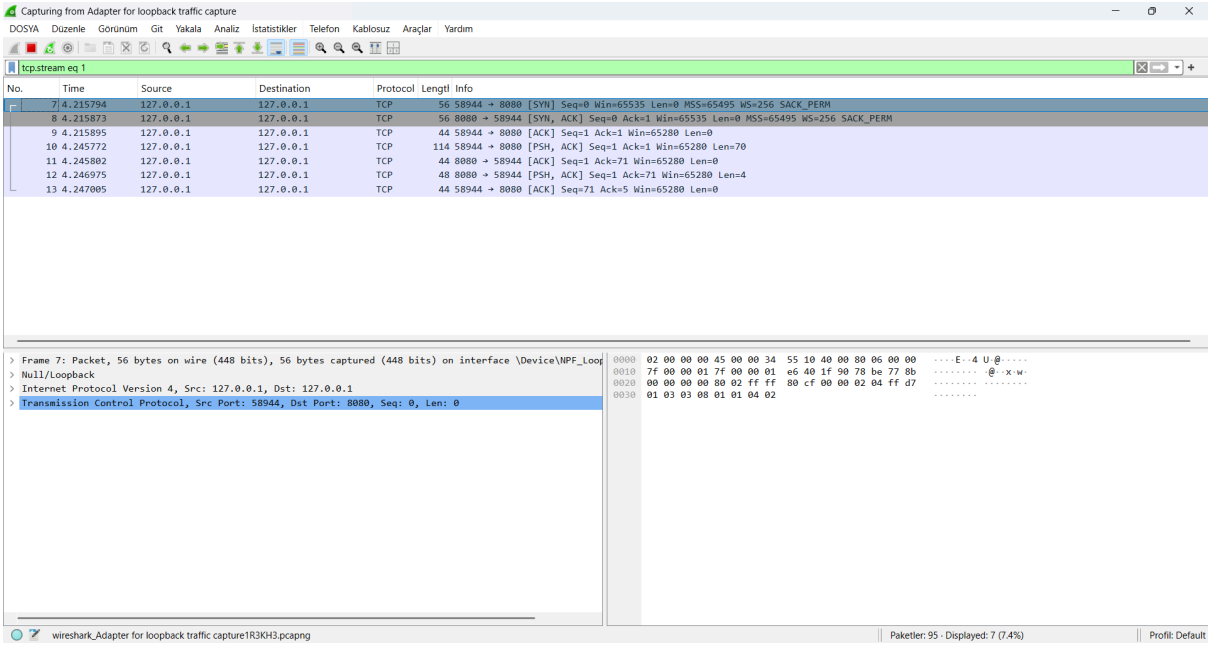
Şifrelemeler:

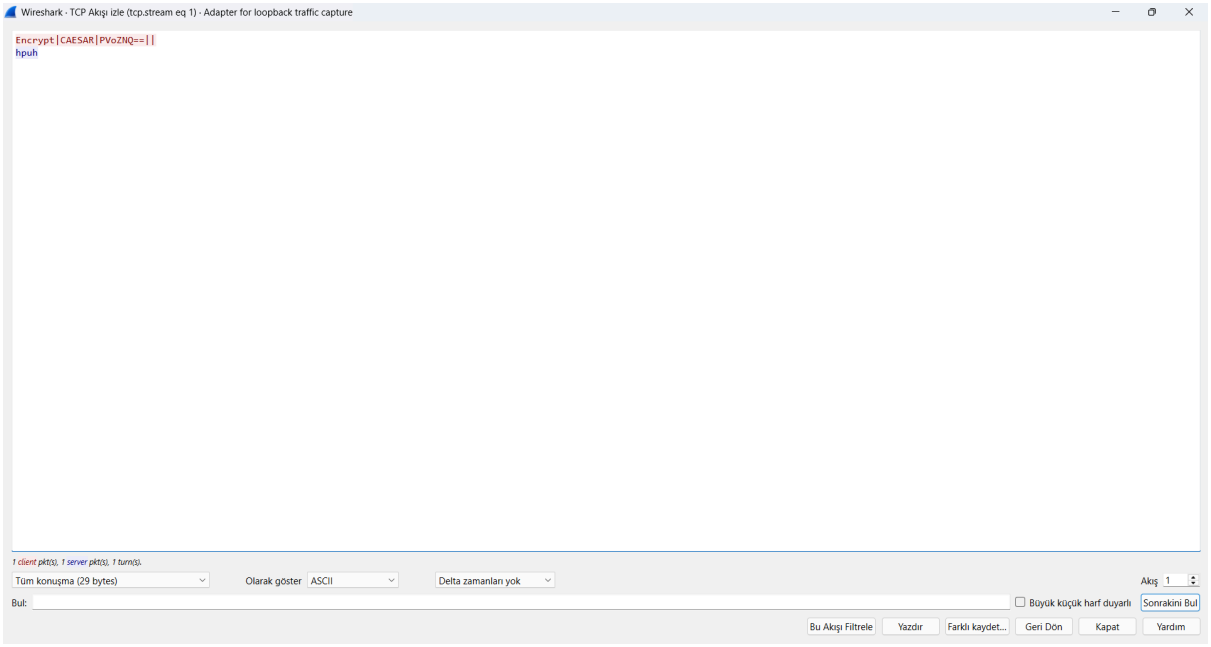


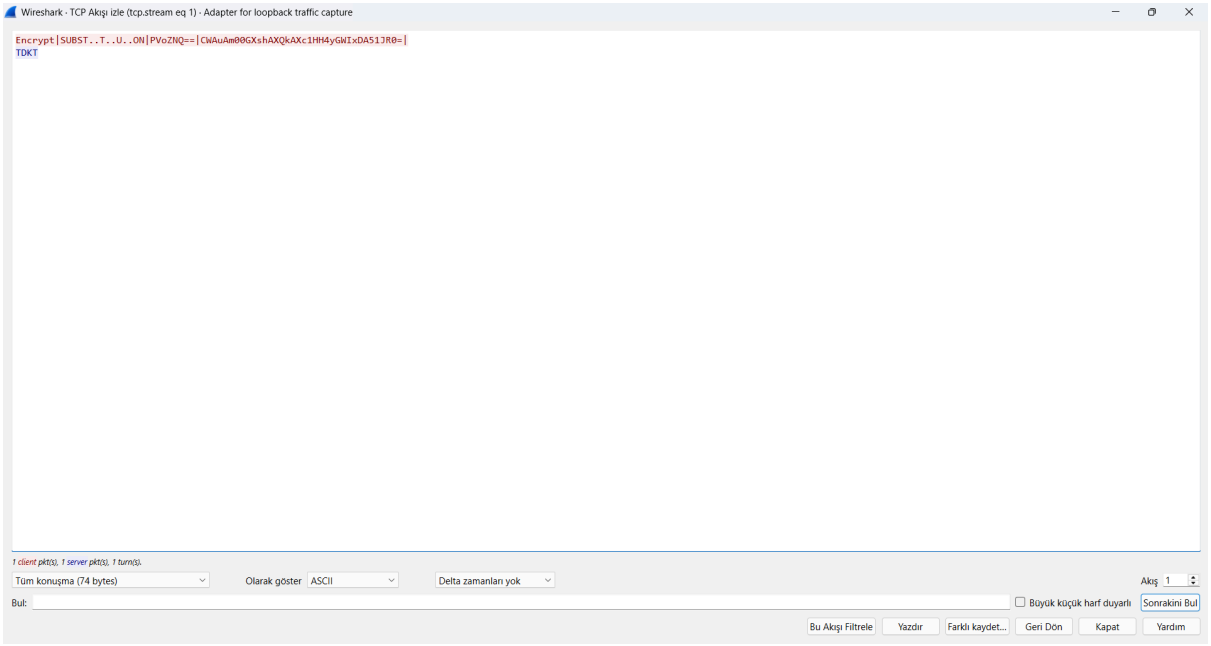

Encryptor
Encrypt
Metin :
emre
Key:
QWERTYUIOPASDFGHJKLZXCVBNM
Opsiyonel IV:
Çıktı :
TDKT
Sezar
Vigenere
Substitiuion
Affine
ROTA
Columnar
Polybius
Pigpen
Hill
Tren Rayı
AES
DES
Decrypt
Metin :
Key:
Opsiyonel IV:
Çıktı :
Sezar
Vigenere
Substitiuion
Affine
ROTA
Columnar
Polybius
Pigpen
Hill
Tren Rayı
AES
DES
Server'a bağlandı!
Gönderilen: Encrypt|SUBSTİTİUİON|PVoZNQ==|CWAuAm00GXshAXQkAXc1HH4yGWIxDA51JR0=|
Şifreleme Sonucu = TDKT

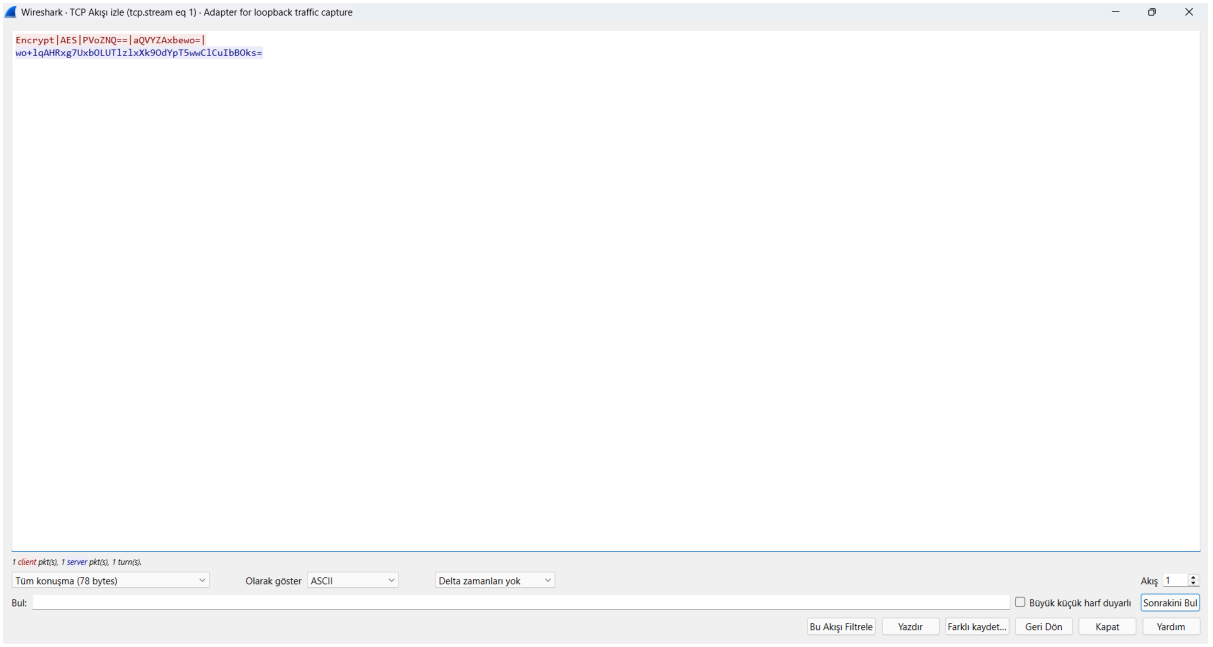
Encryptor
Encrypt
Metin :
emre
Key:
12345678
Opsiyonel IV:
Çıktı :
K1tGvInBZcA2Vx5fwzoDKR27w4PbC
Sezar
Vigenere
Substitiuion
Affine
ROTA
Columnar
Polybius
Pigpen
Hill
Tren Rayı
AES
DES
Decrypt
Metin :
Key:
Opsiyonel IV:
Çıktı :
Sezar
Vigenere
Substitiuion
Affine
ROTA
Columnar
Polybius
Pigpen
Hill
Tren Rayı
AES
DES
Server'a bağlandı!
Gönderilen: Encrypt|AES|PVoZNQ==|aQVYZAxbewo=|
Şifreleme Sonucu = K1tGvInBZcA2Vx5fwzoDKR27w4PbQVIfiBZBZipmwQY=

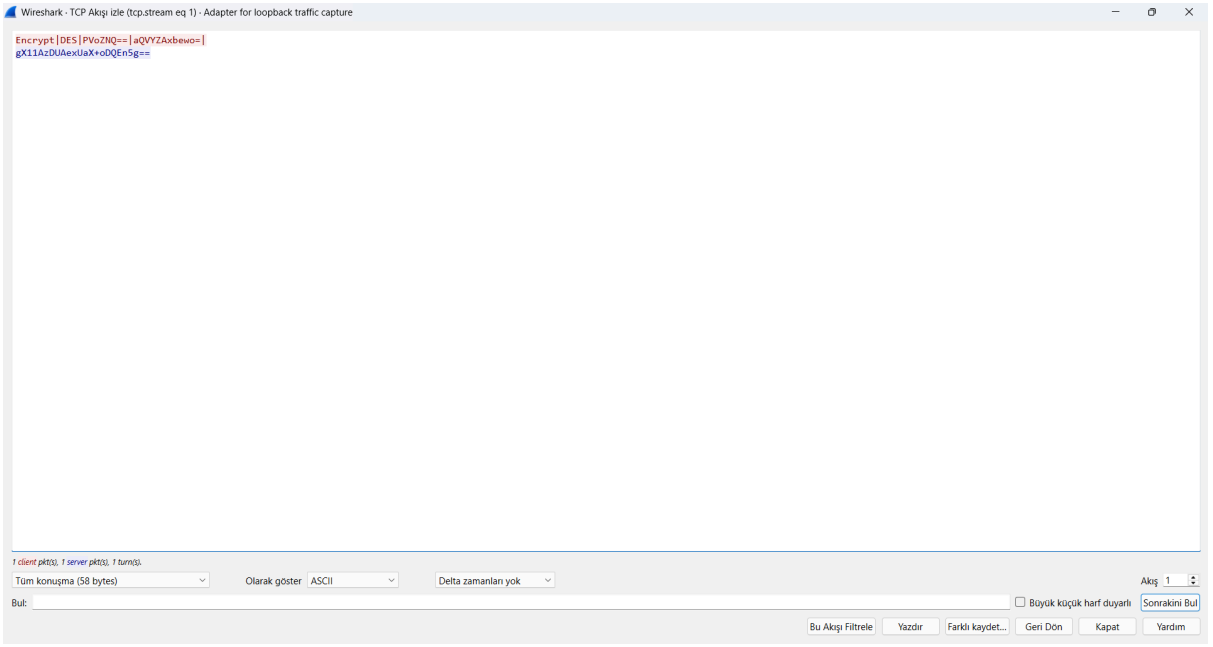

Encryptor
Encrypt
Metin :
emre
Key:
12345678
Opsiyonel IV:
Çıktı :
v3fE+2W8k999DcdDD51iAA==
Sezar
Vigenere
Substitiuion
Affine
ROTA
Columnar
Polybius
Pigpen
Hill
Tren Rayı
AES
DES
Decrypt
Metin :
Key:
Opsiyonel IV:
Çıktı :
Sezar
Vigenere
Substitiuion
Affine
ROTA
Columnar
Polybius
Pigpen
Hill
Tren Rayı
AES
DES
Server'a bağlandı!
Gönderilen: Encrypt|DES|PVoZNQ==|aQVYZAxbewo=|
Şifreleme Sonucu = v3fE+2W8k999DcdDD51iAA==


Capturing from Adapter for loopback traffic capture
DOSYA Düzenle Görünüm Git Yakala Analiz İstatistikler Telefon Kablosuz Araçlar Yardım
tcp.stream eq 1
No. Time Source Destination Protocol Lengtl Info
7 4.215794 127.0.0.1 127.0.0.1 TCP 56 58944 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
8 4.215873 127.0.0.1 127.0.0.1 TCP 56 8080 → 58944 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
9 4.215895 127.0.0.1 127.0.0.1 TCP 44 58944 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
10 4.245772 127.0.0.1 127.0.0.1 TCP 114 58944 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=70
11 4.245802 127.0.0.1 127.0.0.1 TCP 44 8080 → 58944 [ACK] Seq=1 Ack=71 Win=65280 Len=0
12 4.246975 127.0.0.1 127.0.0.1 TCP 48 8080 → 58944 [PSH, ACK] Seq=1 Ack=71 Win=65280 Len=4
13 4.247005 127.0.0.1 127.0.0.1 TCP 44 58944 → 8080 [ACK] Seq=71 Ack=5 Win=65280 Len=0
Frame 7: Packet, 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_Loop
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 58944, Dst Port: 8080, Seq: 0, Len: 0
0000 02 00 00 00 45 00 00 34 55 10 40 00 80 06 00 00
0010 7f 00 00 01 7f 00 00 01 e6 40 1f 90 78 be 77 8b
0020 00 00 00 00 80 02 ff ff 80 cf 00 00 02 04 ff d7
0030 01 03 03 08 01 01 04 02
wireshark_Adapter for loopback traffic capture1R3KH3.pcapng
Paketler: 95 · Displayed: 7 (7.4%)
Profil: Default

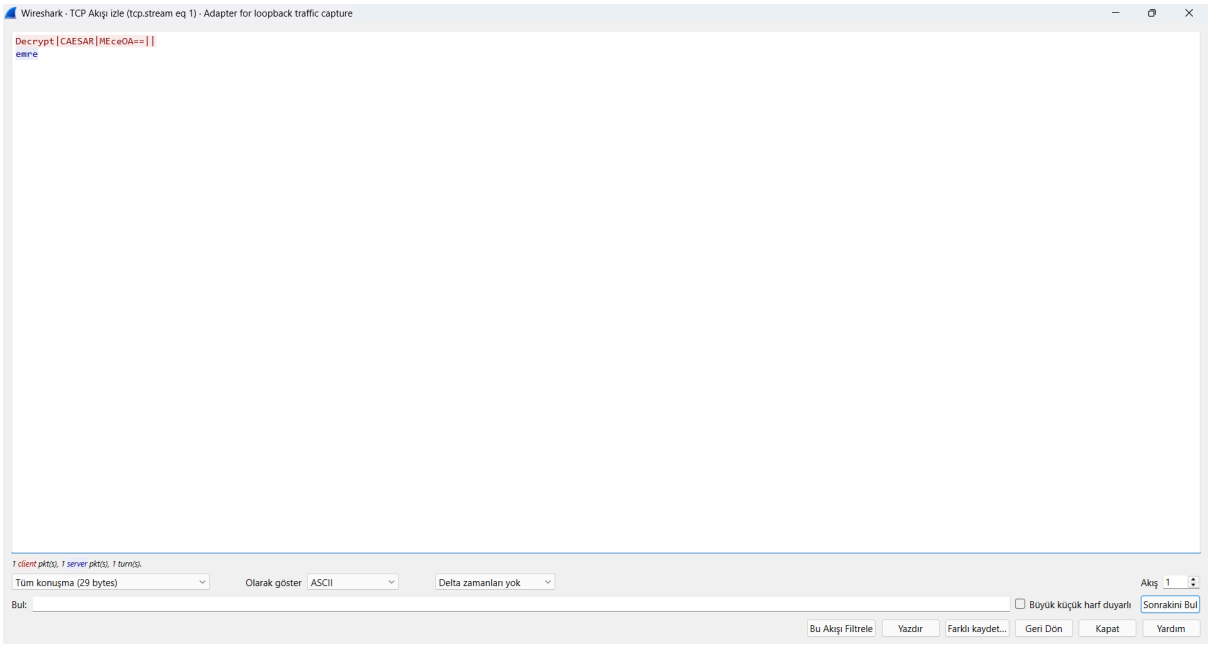

Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Encrypt|CAESAR|PVoZNQ==||
hpuh
1 client pkt(s), 1 server pkt(s), 1 turn(s).
Tüm konuşma (29 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış 1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım

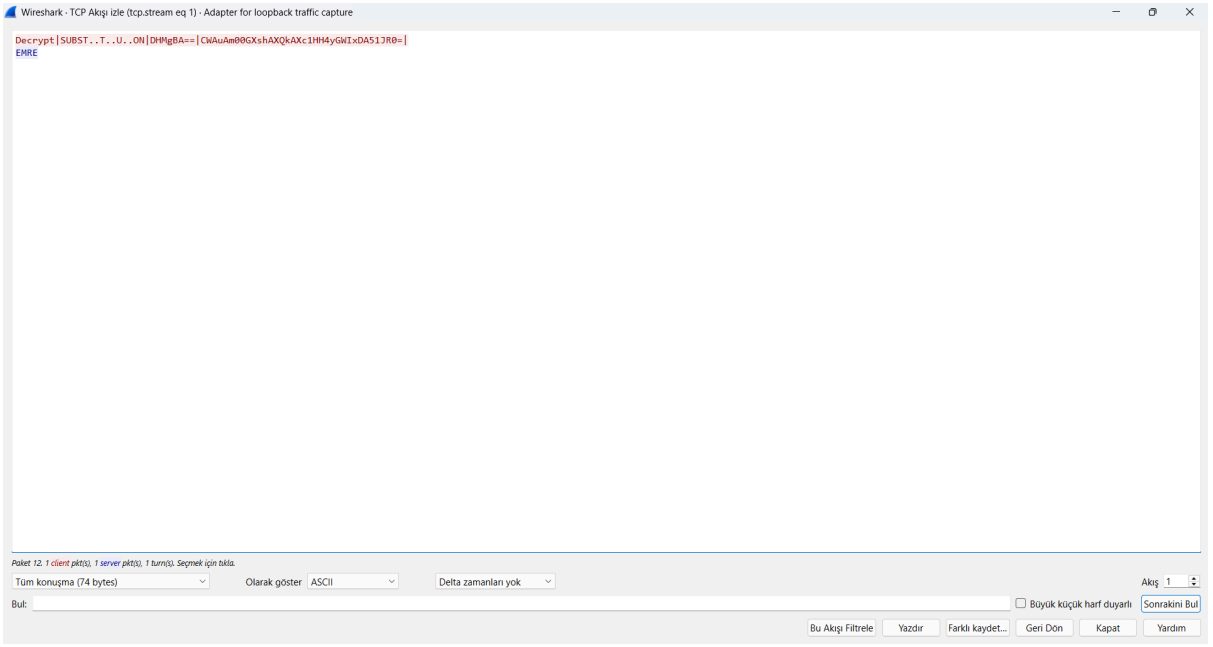

Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Encrypt|SUBST..T..U..ON|PVoZNQ==|CWAuAm00GXshAXQkAXc1HH4yGWIxDA51JR0=|
TDKT
1 client pkt(s), 1 server pkt(s), 1 turn(s).
Tüm konuşma (74 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış 1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım

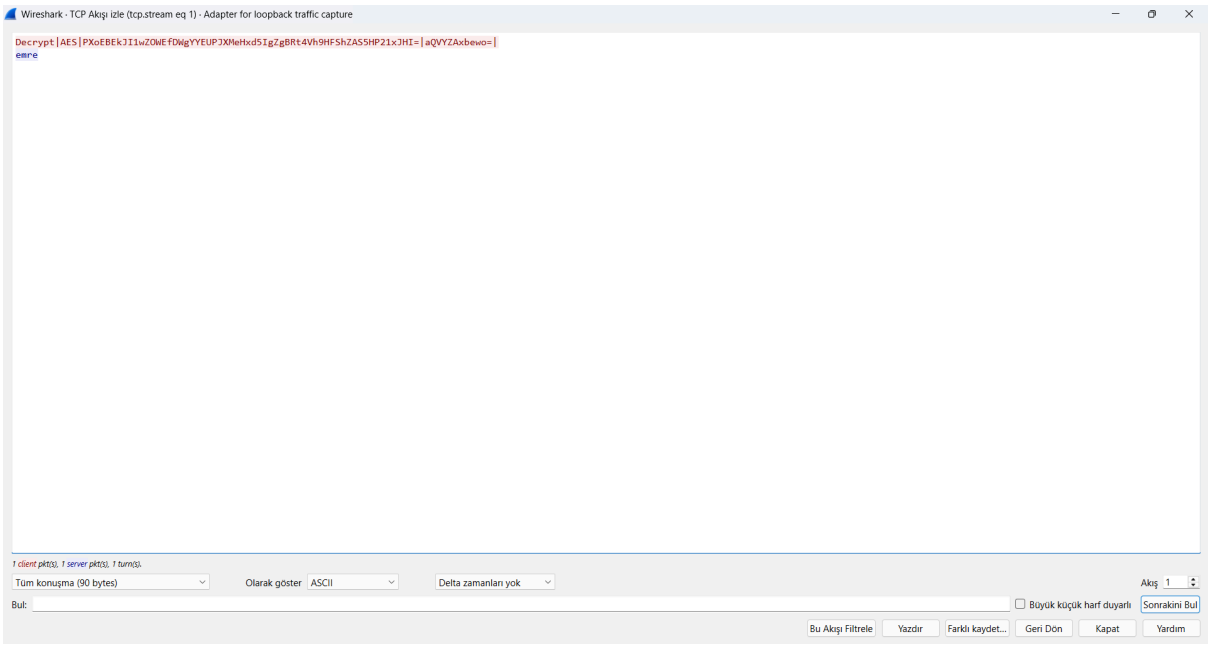

Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Encrypt|AES|PVoZNQ==|aQVYZAxbewo=|
wo+lqAHRxg7UxbOLUTlz1xXk9OdYpT5wwClCuIbBOks=
1 client pkt(s), 1 server pkt(s), 1 turn(s).
Tüm konuşma (78 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış
1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım

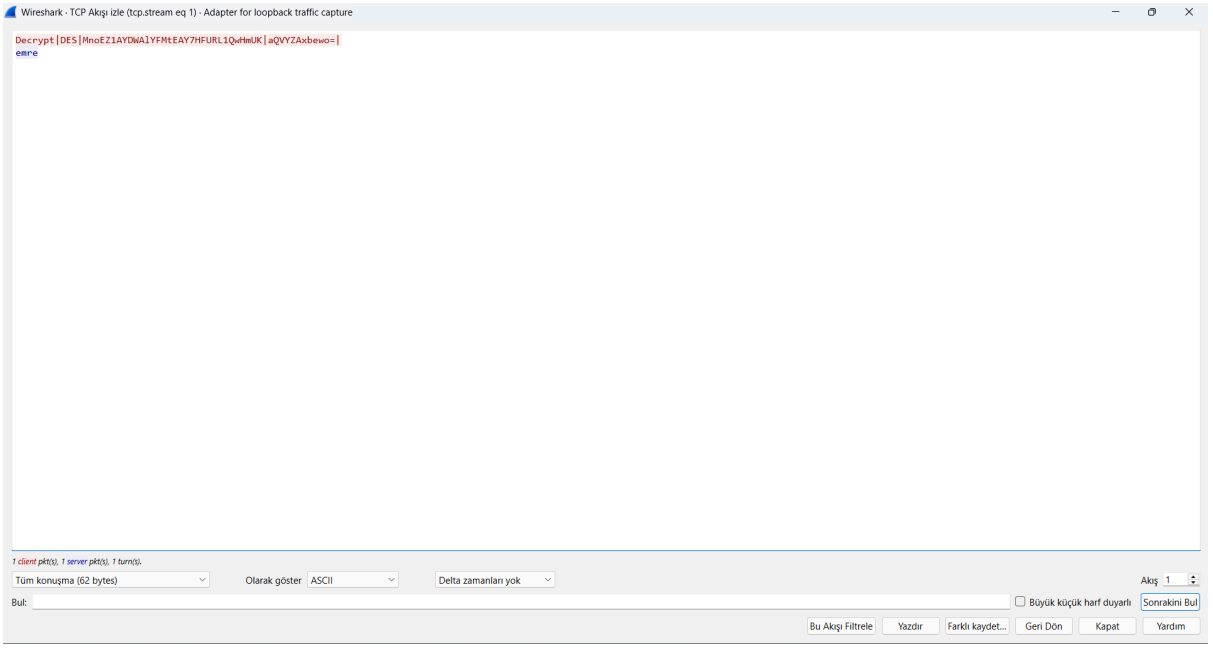

Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Encrypt|DES|PVoZNQ==|aQVYZAxbewo=|
gX11AzDUAexUaX+oDQEn5g==
1 client pkt(s), 1 server pkt(s), 1 turn(s).
Tüm konuşma (58 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış 1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım

Çözmeler:


Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Decrypt|CAESAR|MEceOA==||
emre
1 client pkt(s), 1 server pkt(s), 1 turn(s).
Tüm konuşma (29 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış 1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım


Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Decrypt|SUBST..T..U..ON|DHMgBA==|CWAuAm00GXshAXQkAXc1HH4yGWIxDA51JR0=|
EMRE
Paket 12. 1 client pkt(s), 1 server pkt(s), 1 turn(s). Seçmek için tıkla.
Tüm konuşma (74 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış 1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım


Wireshark · TCP Akışı izle (tcp.stream eq 1) · Adapter for loopback traffic capture
Decrypt|AES|PXoEBEkJI1wZOWEfDWgYYEUPJXMeHxd5IgZgBRt4Vh9HFShZAS5HP21xJHI=|aQVYZAxbewo=|
emre
1 client pkt(s), 1 server pkt(s), 1 turn(s).
Tüm konuşma (90 bytes)
Olarak göster
ASCII
Delta zamanları yok
Akış 1
Bul:
Büyük küçük harf duyarlı
Sonrakini Bul
Bu Akışı Filtrele
Yazdır
Farklı kaydet...
Geri Dön
Kapat
Yardım



---

5. Sonuç ve Değerlendirme

Bu proje kapsamında, kriptoloji biliminin tarihsel gelişiminden günümüze kadar uzanan temel şifreleme yöntemleri incelenmiş ve **C# / .NET** platformu üzerinde güvenli bir **İstemci-Sunucu (Client-Server)** haberleşme sistemi başarıyla geliştirilmiştir.

Çalışma sonucunda elde edilen temel kazanımlar ve teknik çıktılar şunlardır:

1. **Kapsamlı Algoritma Entegrasyonu:** Proje, sadece tek bir döneme odaklanmak yerine kriptolojiyi bir bütün olarak ele almıştır. Tarihsel süreçte kullanılan **Sezar, Vigenere, Yerine Koyma (Substitution), Affine, Rota, Sütunlu Yer Değiştirme, Polybius, Pigpen, Hill ve Tren Rayı** gibi klasik algoritmalar ile günümüzün modern standartları olan **AES** ve **DES** algoritmaları tek bir arayüz üzerinde başarıyla birleştirilmiştir. Bu çeşitlilik, algoritmaların çalışma mantıklarının (harf değiştirme, blok karıştırma, matris çarpımı vb.) karşılaştırmalı olarak analiz edilmesine olanak sağlamıştır.
2. **Modern ve Klasik Yöntemlerin Analizi:** Modern blok şifreleyiciler (AES, DES) ile klasik yöntemler arasındaki yapısal farklar, güvenlik seviyeleri ve anahtar yönetim mekanizmaları (statik anahtarlar vs. dinamik bloklar) uygulama üzerinden pratik olarak gözlemlenmiştir. Özellikle DES ve AES gibi blok şifreleyicilerde karşılaşılan anahtar uzunluğu ve padding gereksinimleri, geliştirilen dinamik algoritmalarla sorunsuz bir şekilde yönetilmiştir.
3. **Ağ Güvenliği ve Tünelleme:** Verinin sadece uygulama katmanında değil, iletim katmanında (Transport Layer) da korunması hedeflenmiştir. Geliştirilen **XOR Cipher tabanlı Tünelleme Protokolü** sayesinde, klasik veya modern fark etmeksizin tüm şifreli veriler ağa çıkmadan önce ikinci bir güvenlik katmanından geçirilmiştir. Bu sayede ağ trafiği (Wireshark vb. ile) dinlense dahi, saldırganların iletişimin içeriğine ve kullanılan anahtarlara erişmesi engellenmiştir.

4. **Mimari Yetkinlik:** Proje, **SOLID** prensiplerine uygun, **Arayüz (Interface)** tabanlı ve **Asenkron (Async/Await)** çalışan modüler bir mimaride tasarlanmıştır. Bu esnek yapı sayesinde, sisteme gelecekte yeni bir şifreleme algoritması eklemek, mevcut kod yapısını bozmadan mümkün hale getirilmiştir.

Sonuç olarak; bu proje ile ders kapsamında talep edilen şifreleme algoritmalarının, TCP tabanlı güvenli bir ağ mimarisi üzerinde kurgulanan yazılım sistemine doğru ve hatasız bir şekilde entegrasyonu sağlanmıştır.