CLİENT – SERVER UYGULAMASI

1. Projenin Amacı ve Kapsamı

Bu projenin temel amacı, Kriptoloji dersi kapsamında öğrenilen şifreleme algoritmalarının, gerçek bir ağ uygulaması üzerinde pratik uygulamasını göstermektir. Proje, Python programlama dili kullanılarak geliştirilen bir İstemci-Sunucu (Client-Server) mimarisine dayanmaktadır. Uygulama, iki uç nokta arasındaki metin tabanlı iletişimi, seçilen kriptografik yöntemle şifreleyerek ağ üzerindeki dinlemelere (packet sniffing) karşı verinin gizliliğini sağlamayı hedefler.

2. Kullanılan Teknolojiler ve Kütüphaneler

- **Python 3.7+ gereklidir.**
- **Tüm şifrelemeler standart kütüphaneleri kullanır. Herhangi bir kütüphane eklemenize gerek yoktur.**

3. Yazılım Mimarisi ve Kod Yapısı

3.1. Şifreleme Yönetimi (Encryption)

Uygulamanın kalbini chipers.py dosyası oluşturmaktadır. Bu sınıf, farklı şifreleme algoritmalarını (AES, DES, Vigenere, Hill, vb.) tek bir çatı altında toplar ve istemci/sunucu kodunun karmaşıklığını azaltır.

Kod yapısı incelendiğinde **Strategy Design Pattern** (Strateji Tasarım Deseni) benzeri bir yapı kurulduğu görülmektedir. encrypt ve decrypt metotları, method parametresine göre ilgili algoritmayı dinamik olarak seçer.

Desteklenen Algoritmalar:

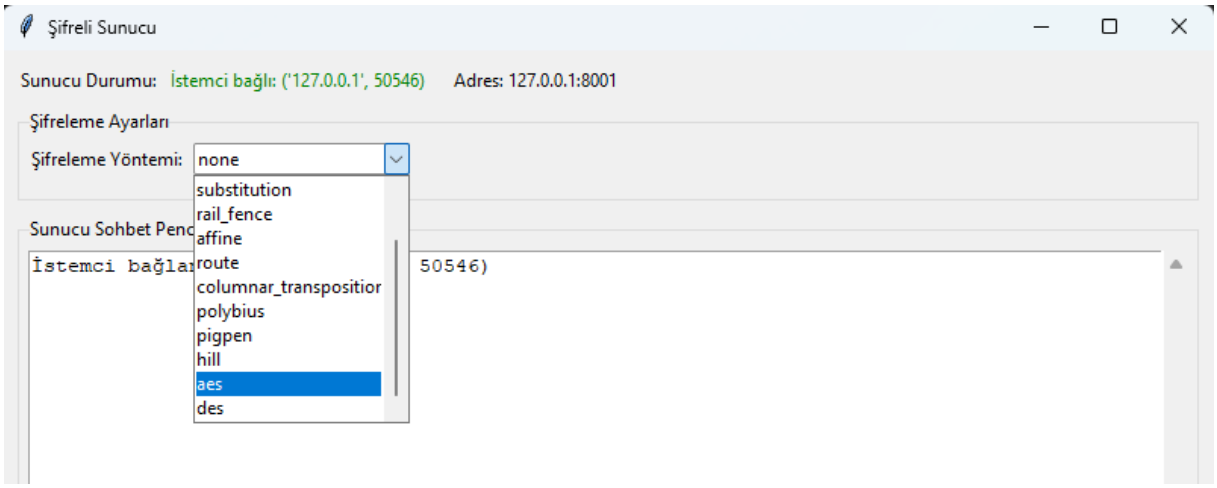
- **Modern Şifreleme:** AES (Advanced Encryption Standard), DES.
- **Klasik Şifreleme:** Caesar, Vigenere, Affine, Rail Fence, Route, Columnar Transposition, Polybius, Pigpen, Hill, Substitution.

3.2. Haberleşme Protokolü (JSON Yapısı)

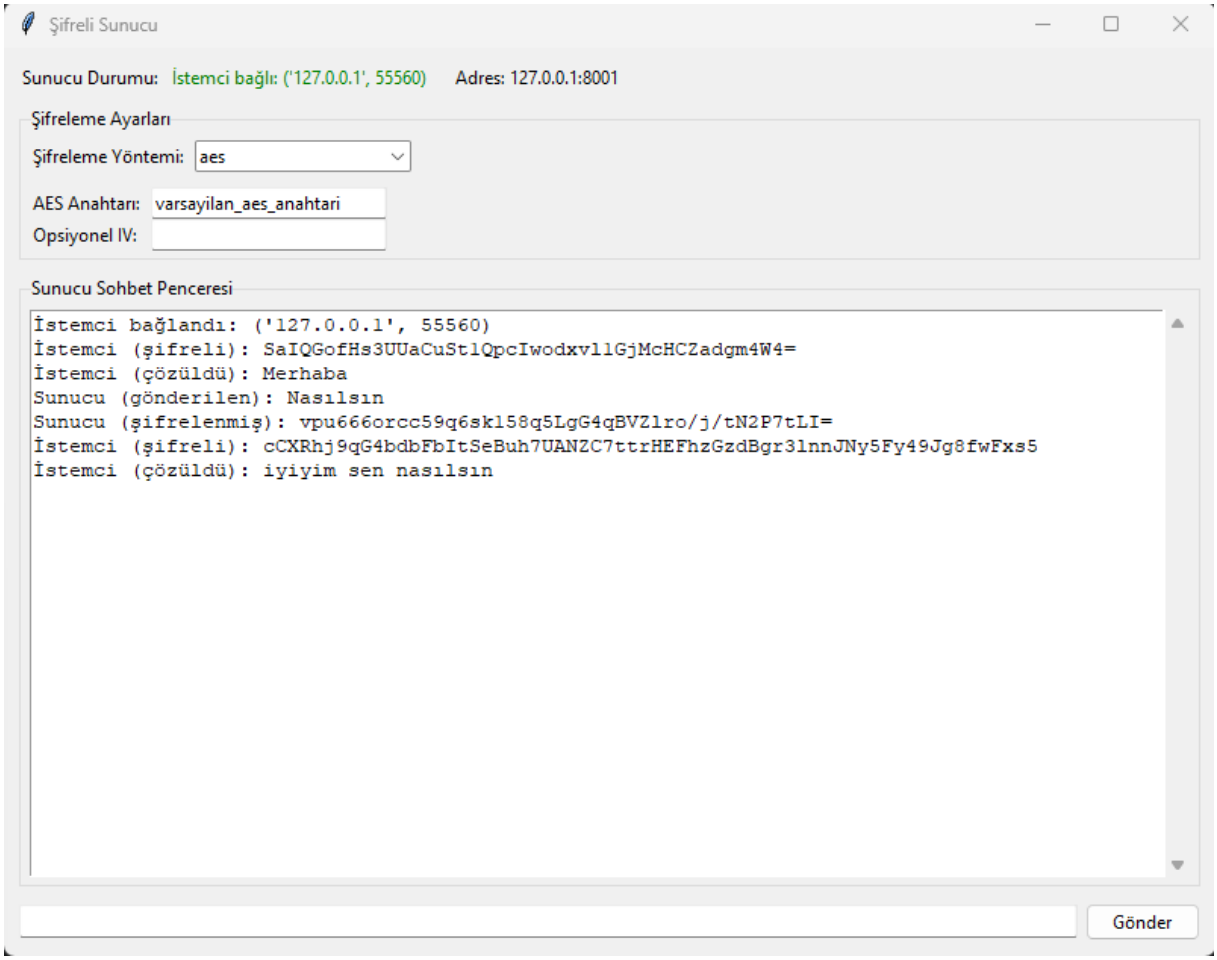
Wireshark analizlerinden görüldüğü üzere (Diğer sayfalarda mevcuttur), sistem ham metin göndermek yerine yapılandırılmış bir JSON formatı kullanmaktadır. Bu format şunları içerir:

- message: Şifrelenmiş metin (Ciphertext).
- method: Kullanılan şifreleme algoritması (örn: "aes", "vigenere").
- params: Şifre çözme için gerekli parametreler (örn: "key", "iv").

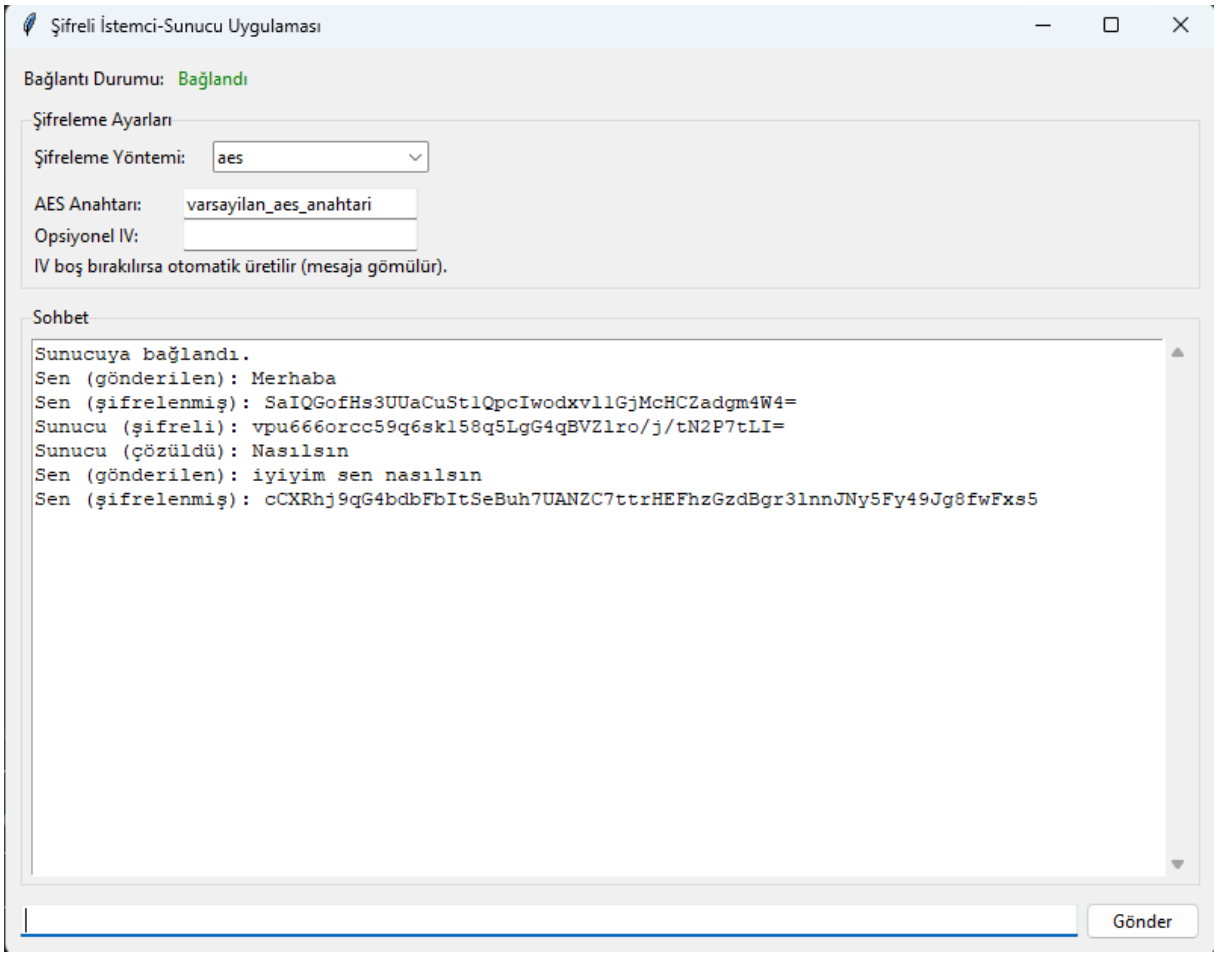
Projenin GUI örnekleri:



- Sunucu yazılan mesajı seçilen şifreleme metoduna göre şifreleyip istemciye gönderir.



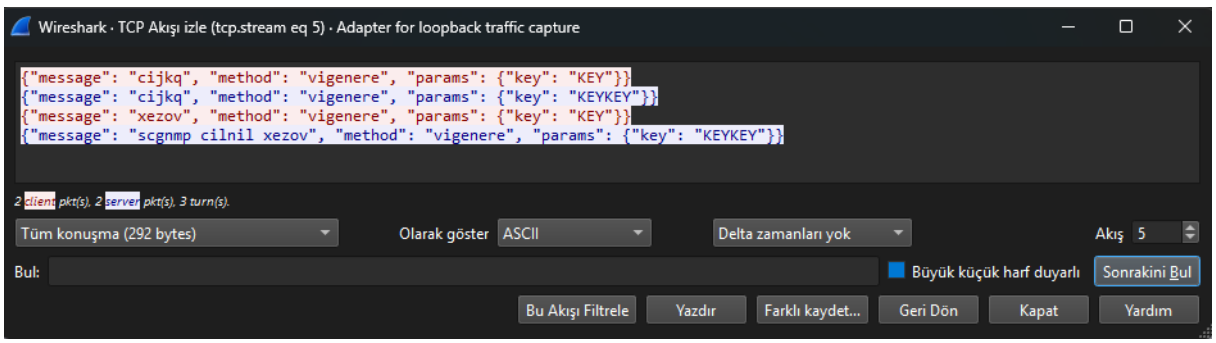
- Şifrelenmiş mesajlar rapor için çıktıları görebilmek amacıyla kullanıcıya verilmiştir.
- AES şifreleme mantığı kod yapısında düzenli bir şekilde yazılmıştır. (S-BOX dahil)



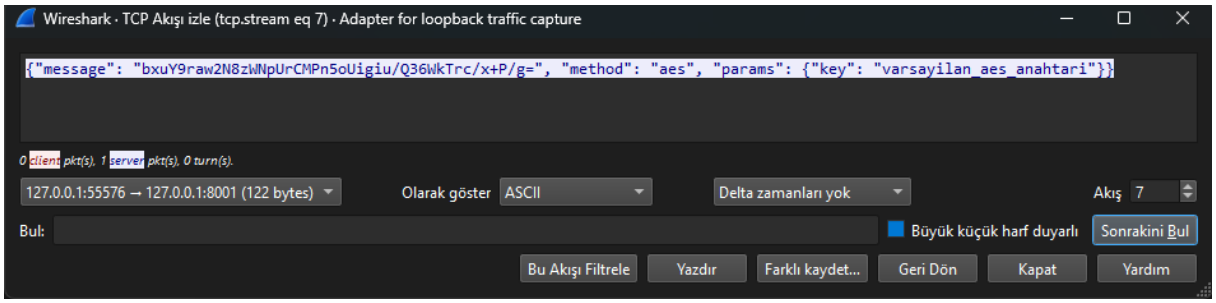
- İstemci sunucudan gelen şifreli datayı decryption işlemine tabii tutup şifreyi çözer ve kullanıcıya verir.

4. Test Sonuçları ve Ağ Analizi

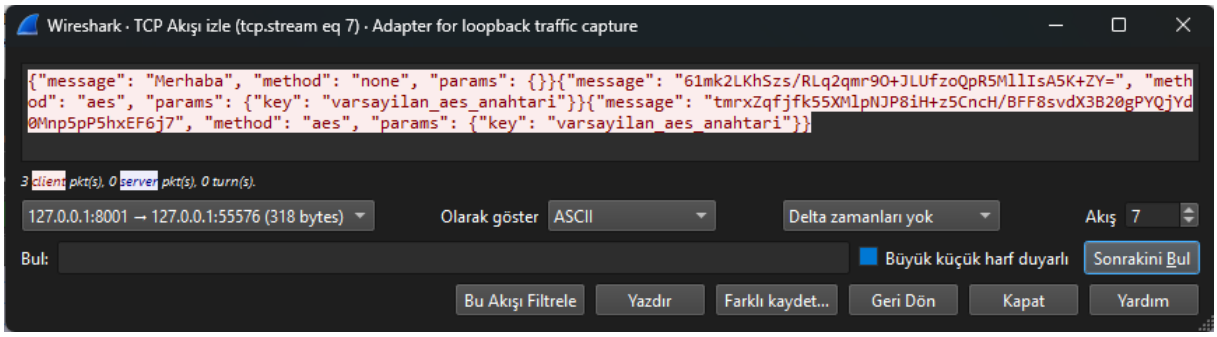
Bu bölümde, uygulamanın çalışır haldeki görüntüleri ve Wireshark ile yapılan ağ trafiği analizleri incelenmiştir.



Bu görüntüde sıradan bir mesajlaşma izlenmiştir.
Selam – Selam – Naber – İyidir senden naber



- Burada şifrelenmiş dataların { “message”:…} "method": "aes", "params": {"key":…} şeklinde gittiği gösterilmiştir.



- İlk mesaj olan “Merhaba” metotların seçilmediği taktirde saf metin gönderdiğini kanıtlamaktadır.
- Metodlar seçildiğinde yazılan metni metoda göre şifreleyip sunucuya gönderdiği izlenmiştir.

4.2. Wireshark ile Ağ Trafiği Analizi

Uygulamanın güvenliğini doğrulamak için "Loopback" arayüzü dinlenmiş ve TCP paketleri yakalanmıştır.

Analiz 1: AES Şifreleme Trafiği

Figür 3 (Wireshark Çıktısı): Ağ üzerinde yakalanan paket içeriği şu şekildedir: {"message": "61mk2LKhSzs/...", "method": "aes", "params": {"key": "varsayilan_aes_anahtari"}}

Bulgular:

1. **Gizlilik:** "Merhaba" gibi açık metinler (plaintext) ağda **görünmemektedir**. Saldırgan sadece anlamsız karakter yığınları (ciphertext) görmektedir.
2. **Bütünlük:** JSON yapısı, mesajın hangi yöntemle şifrelendiğini alıcıya bildirerek senkronizasyon sağlar.

5. Sonuç

Bu proje ile, teorik olarak öğrenilen kriptoloji algoritmalarının çalışan bir yazılım sistemine entegrasyonu başarıyla tamamlanmıştır.

1. **Başarım:** İstemci ve sunucu sorunsuz bir şekilde haberleşmiş ve veri aktarmıştır.
2. **Güvenlik:** Wireshark analizleri, verilerin "clear-text" (açık metin) olarak gitmediğini, aradaki bir saldırganın (Man-in-the-Middle) mesaj içeriğini doğrudan okuyamayacağını kanıtlamıştır.
3. **Esneklik:** Geliştirilen modüler yapı sayesinde sisteme yeni şifreleme algoritmaları kolayca eklenebilir durumdadır.