KRİPTOGRAFİ VE AĞ GÜVENLİĞİ
AES – DES – RSA Kullanarak İstemci–Sunucu Şifreleme Sistemi Analizi

Öğrenci Bilgileri
- **Ad Soyad: Hüseyin Bünyamin Gülme**
- **Öğrenci No: 439591**
- **GitHub Deposu: https://github.com/huseyingulme/Kriptoloji**

1. Giriş ve Projenin Amacı

Bu çalışmanın amacı, modern ağ güvenliği yaklaşımlarını uygulamalı olarak incelemek ve simetrik ile asimetrik şifreleme algoritmalarının gerçek zamanlı istemci–sunucu haberleşmesinde nasıl birlikte kullanıldığını göstermektir.
Bu kapsamda geliştirilen sistem, AES-128, DES ve RSA algoritmalarını destekleyen, TCP tabanlı güvenli veri iletimi sağlayan bir istemci–sunucu mimarisi sunmaktadır. Proje hem kriptografik kütüphaneler kullanılarak hem de manuel (kütüphanesiz) implementasyon yaklaşımıyla çalışabilecek şekilde tasarlanmıştır.
Çalışmanın temel hedefleri şunlardır:
- **Simetrik (AES, DES) ve asimetrik (RSA) şifreleme algoritmalarını uygulamalı olarak kullanmak**
- **RSA'nın anahtar dağıtımında, AES/DES'in ise veri gizliliğinde nasıl konumlandığını göstermek**
- **Şifreli ağ trafiğini Wireshark ile analiz ederek paket yapılarını incelemek**
- **Kütüphaneli ve manuel şifreleme yöntemleri arasındaki farkları gözlemlemek**

2. Sistem Mimarisi ve Genel Yapı

Sistem, Python programlama dili kullanılarak geliştirilmiş olup grafiksel arayüz için Tkinter kütüphanesinden yararlanılmıştır. Mimari yapı üç temel bileşenden oluşmaktadır:
- **İstemci Uygulaması**
- **Sunucu Uygulaması**
- **Şifreleme ve Paket İşleme Katmanı**

2.1. İstemci–Sunucu İletişim Protokolü

İstemci ve sunucu arasındaki iletişim, güvenilir bir taşıma protokolü olan TCP (Transmission Control Protocol) üzerinden sağlanmaktadır. Gönderilen veriler ham (plain text) olarak iletilmemekte; bunun yerine özel olarak tasarlanmış bir DataPacket yapısı içerisinde şifrelenmiş biçimde gönderilmektedir.
Her paket aşağıdaki bileşenleri içermektedir:
- **Veri uzunluğu**
- **Kullanılan algoritma bilgisi (AES, DES, RSA)**
- **İşlem türü (şifreleme / çözme)**
- **Zaman damgası**
- **Şifrelenmiş veri yükü (payload)**

Bu yapı sayesinde paketler hem analiz edilebilir hem de farklı algoritmalarla dinamik olarak işlenebilir hale gelmiştir.

2.2. ProcessingManager ve Dinamik Algoritma Seçimi

Sunucu tarafında, tüm şifreleme algoritmaları bir Registry yapısında tutulmaktadır.
Merkezi kontrol birimi olan ProcessingManager, istemciden gelen paketin metadata bilgilerini inceleyerek hangi algoritmanın kullanılacağını dinamik olarak belirler.
Bu yaklaşımın avantajları:

- **Yeni bir şifreleme algoritması eklemek için ana kod yapısının değiştirilmesine gerek yoktur**
- **Modüler ve genişletilebilir bir mimari elde edilmiştir**
- **Gerçek dünya protokollerinde (TLS gibi) kullanılan stratejilere benzer bir yapı kurulmuştur**

2.3. Hibrit Şifreleme (Hybrid Encryption) Modu

RSA algoritmasının büyük veri boyutlarında yavaş ve maliyetli olması nedeniyle sistemde Hibrit Şifreleme desteği uygulanmıştır. Bu mod, modern HTTPS/TLS protokollerinde kullanılan yaklaşımın birebir karşılığıdır.

Hibrit modun çalışma adımları:

1. **İstemci tarafında rastgele bir AES anahtarı üretilir**
2. **Asıl veri, bu AES anahtarı ile hızlı bir şekilde şifrelenir**
3. **AES anahtarı, sunucunun RSA Public Key'i ile şifrelenir**
4. **Sunucu, kendi Private Key'i ile AES anahtarını çözer**
5. **Ardından AES ile şifrelenmiş ana veri çözülür**

Bu yöntem sayesinde RSA'nın güvenliği ile AES'in performansı birleştirilmiştir.

3. Kullanılan Kriptografik Algoritmalar

3.1. AES-128 (Advanced Encryption Standard)

AES, 128 bit bloklar üzerinde çalışan güçlü bir simetrik şifreleme algoritmasıdır. Bu projede hem kütüphane tabanlı hem de manuel bir AES-128 implementasyonu gerçekleştirilmiştir.

Manuel AES uygulamasında 10 round'luk yapı kullanılmıştır:

- **SubBytes: S-Box tablosu kullanılarak doğrusal olmayan dönüşüm**
- **ShiftRows: Satırların dairesel kaydırılması**
- **MixColumns: Matematiksel matris çarpımı ile sütun karıştırma**
- **AddRoundKey: Round anahtarının XOR işlemi ile eklenmesi**

Bu yapı sayesinde AES'in iç çalışma mantığı doğrudan deneyimlenmiştir.

3.2. DES (Data Encryption Standard)

DES, 64 bit bloklar üzerinde çalışan ve Feistel yapısını kullanan simetrik bir algoritmadır.

Manuel DES implementasyonunda aşağıdaki bileşenler kodlanmıştır:

- **Initial Permutation (IP)**
- **16 Round Feistel Yapısı**
- **F-Function: Genişletme, XOR ve S-Box işlemleri**
- **Key Schedule: 64 bit anahtardan 16 alt anahtar üretimi**

DES'in günümüzde zayıf kabul edilmesine rağmen, eğitimsel açıdan algoritma yapısını anlamak için oldukça öğretici olduğu gözlemlenmiştir.

3.3. RSA (Rivest–Shamir–Adleman)

RSA, asimetrik şifreleme algoritmalarının en bilinen örneklerinden biridir. Bu projede RSA, veri şifreleme amacıyla değil, anahtar dağıtımı amacıyla kullanılmıştır.

RSA'nın güvenliği,

$$n = p \times q$$

şeklinde oluşturulan büyük asal sayıların çarpanlara ayrılmasının zorluğuna dayanır.

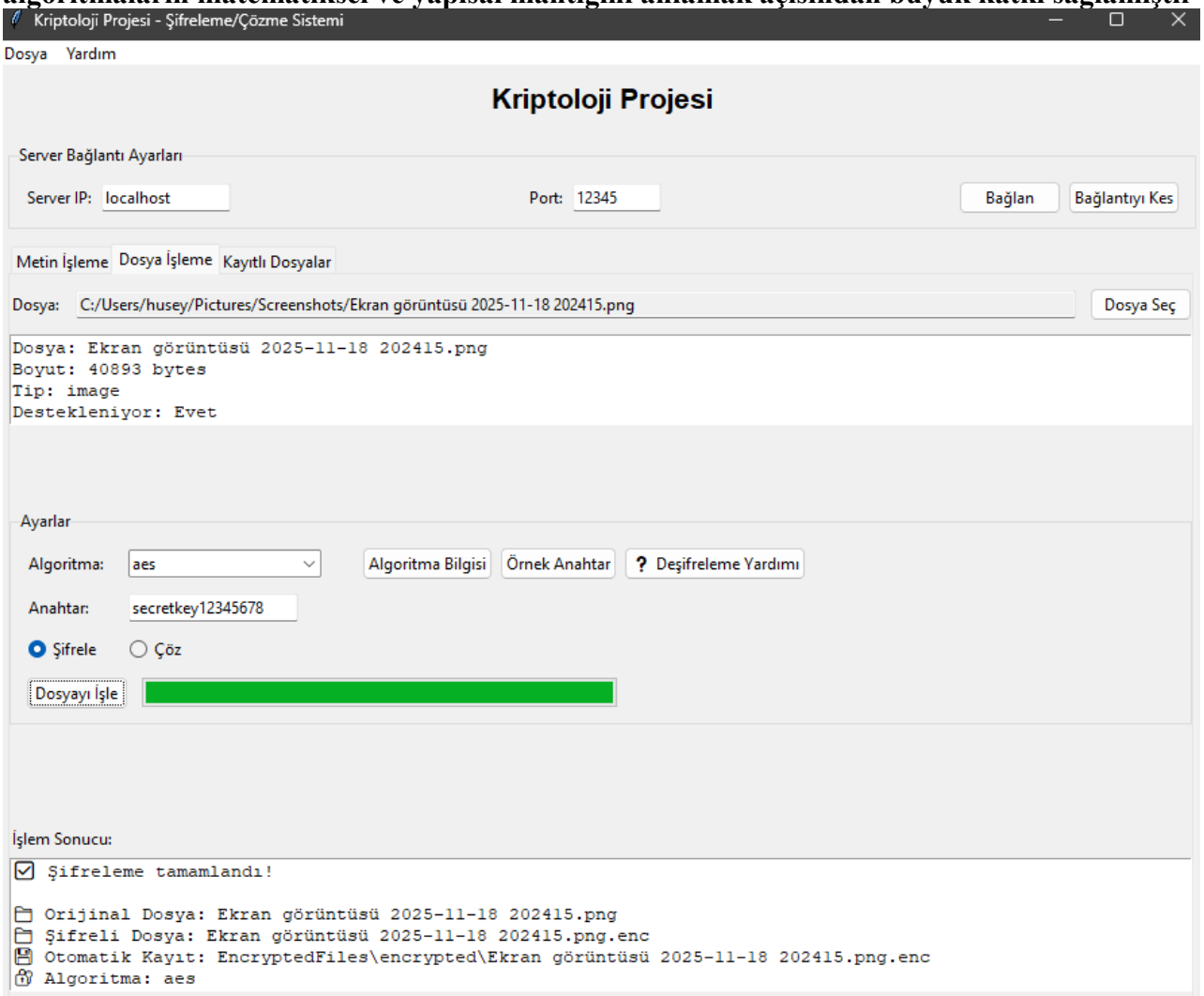
Projede:

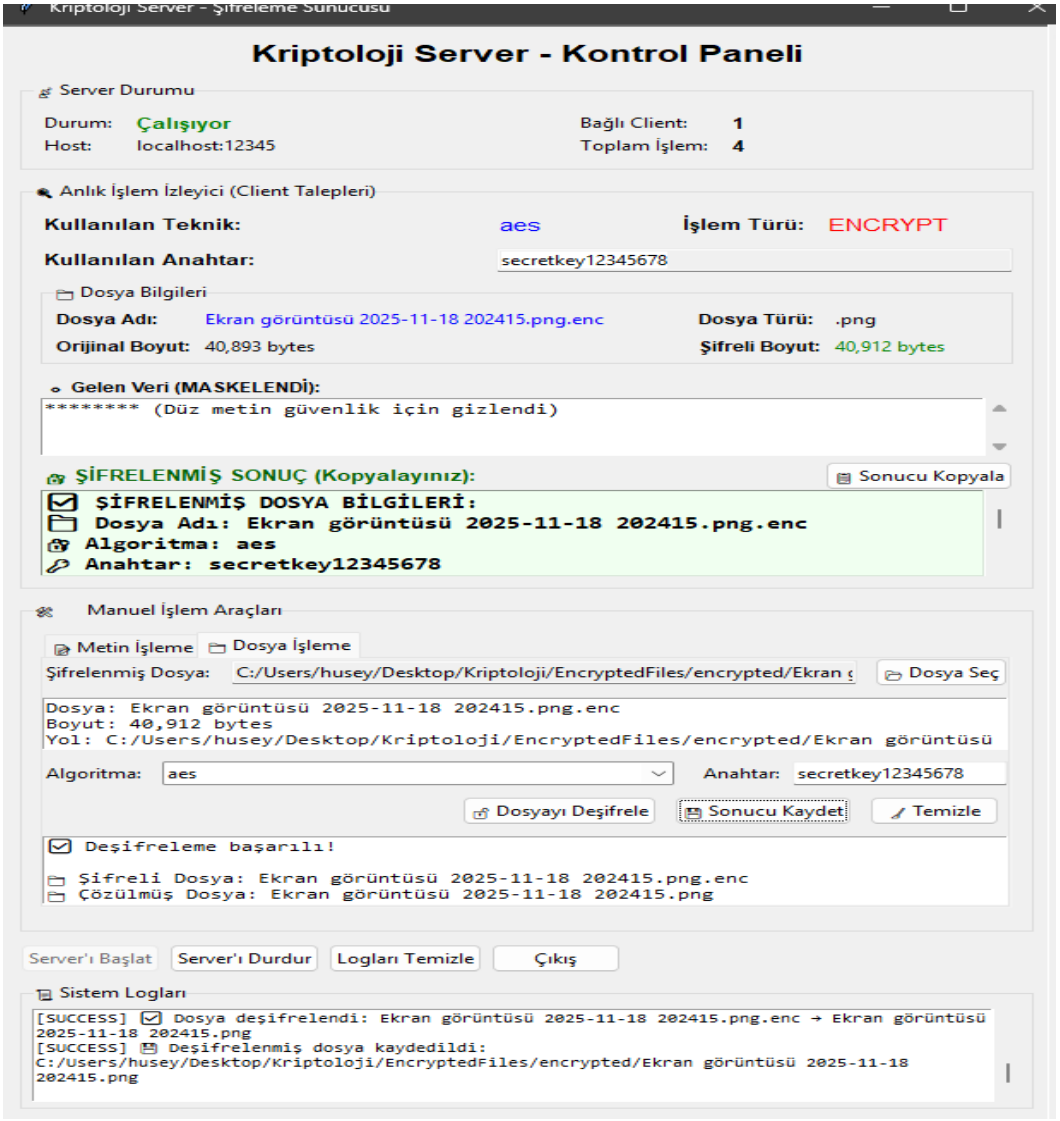
- **1024 bit ve 2048 bit RSA anahtarlarıyla testler yapılmıştır**
- **RSA'nın büyük veri boyutlarında neden verimsiz olduğu deneysel olarak gözlemlenmiştir**
- **Manuel RSA implementasyonu yapılmamış, ancak teorik yapısı raporda açıklanmıştır**

4. Karşılaştırmalı Analiz

Kriter	Simetrik (AES / DES)	Asimetrik (RSA)
Hız	Çok hızlı	Yavaş
Kullanım	Veri şifreleme	Anahtar değişimi
Kaynak Tüketimi	Düşük	Yüksek
Veri Boyutu	Küçük	Anahtar boyutuna eşit

Manuel vs Kütüphaneli:
Kütüphane tabanlı çözümler (PyCryptodome / Hazmat), donanım hızlandırmaları (AES-NI) sayesinde manuel implementasyonlardan çok daha hızlıdır. Ancak manuel kodlama, algoritmaların matematiksel ve yapısal mantığını anlamak açısından büyük katkı sağlamıştır





5. Wireshark Ağ Trafiği Analizi

Bu çalışmada geliştirilen istemci–sunucu uygulaması çalıştırılarak, taraflar arasındaki TCP haberleşmesi Wireshark aracı kullanılarak detaylı biçimde analiz edilmiştir. Paket yakalama işlemi sırasında, şifreleme algoritmalarının ağ trafiği üzerindeki etkileri doğrudan gözlemlenmiştir.

5.1. Veri Gizliliği

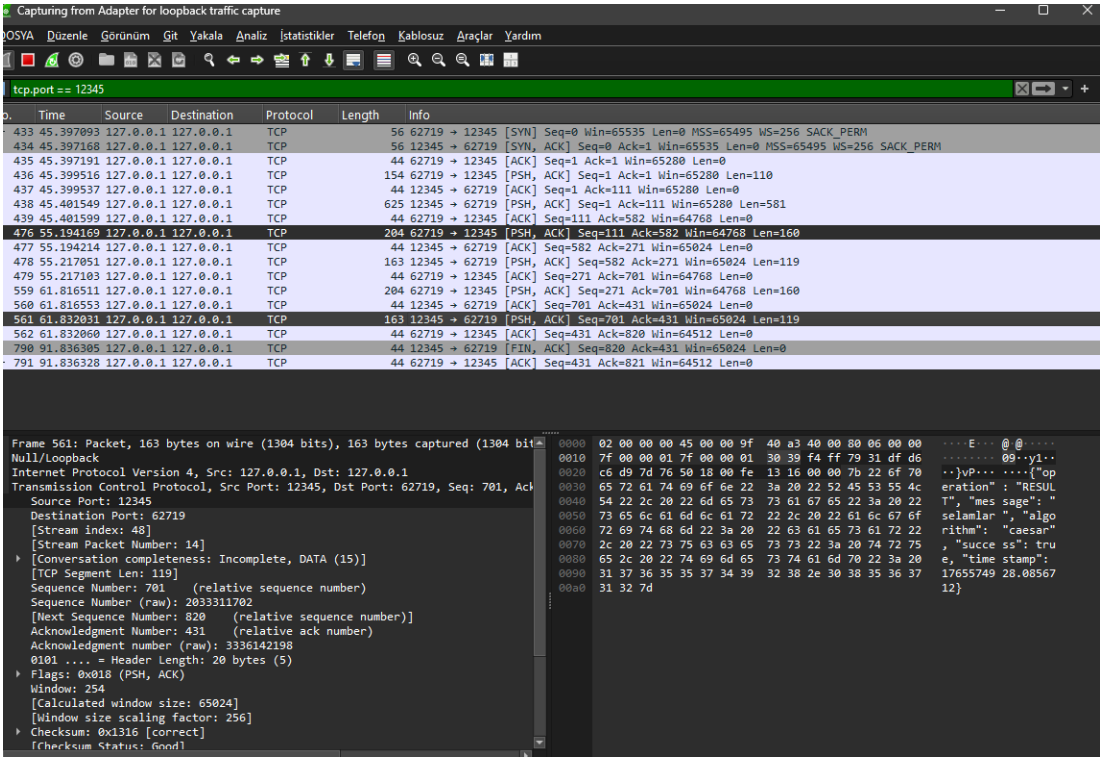
Wireshark üzerinden yakalanan TCP paketlerinin payload (yük) alanı incelendiğinde aşağıdaki bulgular elde edilmiştir:

- **Ağ üzerinden iletilen veriler açık metin (plain text) biçiminde görülmemektedir.**
- **"Merhaba", "Ödev Raporu" gibi kullanıcı tarafından gönderilen okunabilir ifadeler payload alanında yer almamaktadır.**
- **Bunun yerine, anlamsız görünen Hexadecimal ve Base64 formatında veri dizileri bulunmaktadır.**
- **Payload içeriğinin bu şekilde okunamaz olması, verinin istemci tarafında başarılı bir şekilde şifrelendiğini göstermektedir.**
- **Bu durum, ağ trafiğini dinleyen üçüncü şahısların iletilen mesajın içeriğini doğrudan okuyamayacağını kanıtlamaktadır.**
- **Elde edilen sonuçlar, geliştirilen sistemin veri gizliliği açısından güvenli bir haberleşme sağladığını ortaya koymaktadır.**

5.2. Paket Boyutları ve RSA Algoritmasının Etkisi

Wireshark analizinde, kullanılan şifreleme algoritmasına bağlı olarak TCP paket boyutlarında belirgin farklılıklar tespit edilmiştir:

- **AES kullanıldığında:**
  - **10 byte uzunluğundaki bir mesaj, blok şifreleme yapısı nedeniyle padding ve IV eklenerek yaklaşık 16–32 byte aralığında bir payload oluşturmuştur.**
  - **Bu artış, AES'in sabit blok yapısının doğal bir sonucudur ve performans açısından kabul edilebilir düzeydedir.**
- **RSA (2048 bit) kullanıldığında:**
  - **Aynı büyüklükteki bir mesajın şifrelenmesi sonucunda payload boyutu 256 byte olarak gözlemlenmiştir.**
  - **Bunun nedeni, RSA algoritmasının çıktı boyutunun her zaman anahtar uzunluğuna eşit olmasıdır.**
  - **Küçük bir veri dahi şifrelense, RSA tarafından üretilen şifreli çıktı sabit ve büyük boyutlu olmaktadır.**
- **Bu durum, RSA'nın büyük veri bloklarının doğrudan şifrelenmesi için verimsiz olduğunu göstermektedir.**
- **Elde edilen bulgular, RSA'nın pratikte neden yalnızca anahtar dağıtımı ve dijital imza gibi işlemler için kullanıldığını açıkça ortaya koymaktadır.**



6. Sonuç ve Kişisel Değerlendirme

Bu proje, şifrelemenin yalnızca veriyi gizlemekten ibaret olmadığını; anahtar yönetimi, protokol tasarımı ve performans dengesi gerektiren çok katmanlı bir süreç olduğunu açıkça ortaya koymuştur.

Manuel AES ve DES implementasyonları sayesinde algoritmaların iç yapısı doğrudan deneyimlenmiş, Wireshark analizi ile teorik bilgilerin ağ katmanlarında nasıl karşılık bulduğu gözlemlenmiştir. Hibrit şifreleme yaklaşımı ise modern internet protokollerinin temel mantığını anlamak açısından son derece öğretici olmuştur.