

CLIENT–SERVER ŞİFRELEME SİSTEMİ

Numara: 439591

İsim: Hüseyin Bünyamin Gülme

GitHub Repo: <https://github.com/huseyingulme/Kriptoloji>

1. Projenin Amacı ve Kapsamı

Bu projenin amacı, klasik ve modern kriptografi algoritmalarının çalışma prensiplerini uygulamalı olarak incelemek ve bu algoritmaları istemci–sunucu (Client–Server) mimarisi üzerinde çalışan gerçek bir yazılım sistemi içerisinde kullanmaktır. Kriptoloji dersi kapsamında öğrenilen teorik bilgilerin pratikte nasıl uygulandığını göstermek hedeflenmiştir.

Proje, Python programlama dili kullanılarak TCP/IP protokolü üzerinden haberleşen bir istemci–sunucu yapısı şeklinde geliştirilmiştir. Sistem; AES, DES ve RSA gibi modern kriptografik algoritmaların yanı sıra Caesar, Vigenère, Hill ve Playfair gibi klasik şifreleme yöntemlerini de desteklemektedir. Böylece algoritmalar hem güvenlik hem de tarihsel gelişim açısından karşılaştırılabilir hale getirilmiştir.

Uygulamada iki temel yaklaşım kullanılmıştır:

1. Kütüphane Tabanlı Şifreleme:

Python cryptography kütüphanesi kullanılarak AES, DES ve RSA algoritmaları güvenli ve standartlara uygun şekilde uygulanmıştır.

2. Manuel Şifreleme Yaklaşımı:

Klasik şifreleme algoritmaları ve bazı modern algoritmaların temel mantığı manuel olarak kodlanarak algoritmaların iç yapıları daha iyi anlaşılmıştır.

Bu kapsamda veri gizliliği (confidentiality), anahtar yönetimi ve ağ güvenliği konuları uygulamalı olarak ele alınmıştır.

2. Sistem Mimarisi ve Çalışma Akışı

Uygulama, istemci–sunucu mimarisi üzerine kurulmuştur. İstemci kullanıcıdan aldığı verileri sunucuya gönderir, tüm şifreleme ve çözme işlemleri sunucu tarafında gerçekleştirilir.

2.1. Çalışma Akışı

1. İstemci uygulaması başlatılır ve sunucuya TCP bağlantısı kurulur.
2. Kullanıcı, şifrelenecek metni, algoritmayı ve anahtar bilgisini girer.
3. İstemci, bu bilgileri JSON formatında sunucuya gönderir.
4. Sunucu, gelen isteği ayrıştırır ve ilgili algoritmayı çalıştırır.
5. Şifrelenen veya çözülen veri istemciye geri gönderilir.
6. Sonuç kullanıcı arayüzünde görüntülenir.

3. Kullanılan Algoritmalar ve Karşılaştırma

Projede hem simetrik hem de asimetrik şifreleme algoritmaları kullanılmıştır.

3.1. AES (Advanced Encryption Standard)

AES, simetrik blok şifreleme algoritmasıdır ve 128, 192 ve 256 bit anahtar uzunluklarını destekler. Yüksek güvenliği ve performansı nedeniyle modern sistemlerde standart olarak kullanılmaktadır. Projede hem kütüphane tabanlı hem de manuel implementasyonları bulunmaktadır.

3.2. DES (Data Encryption Standard)

DES, 64 bit blok boyutuna ve 56 bit anahtar uzunluğuna sahip eski bir simetrik şifreleme algoritmasıdır. Günümüzde güvenli kabul edilmemektedir ancak eğitim amacıyla blok şifreleme mantığını anlamak için projeye dahil edilmiştir.

3.3. RSA (Rivest–Shamir–Adleman)

RSA, asimetrik bir şifreleme algoritmasıdır ve açık anahtar–özel anahtar yapısına sahiptir. Projede 1024 ila 4096 bit arasında anahtar uzunlukları desteklenmektedir. RSA, büyük verilerin şifrelenmesi yerine anahtar paylaşımı amacıyla kullanılmıştır.

| Özellik | AES | DES | RSA |
|----------------|---|----------|---------------|
| Tür | Simetrik | Simetrik | Asimetrik |
| Anahtar Boyutu | 128–256 bit | 56 bit | 1024–4096 bit |
| Hız | Çok hızlı | Hızlı | Yavaş |
| Kullanım | Veri şifreleme Eğitim/Legacy Anahtar dağıtımı | | |

4. Manuel ve Kütüphane Tabanlı Şifreleme Analizi

4.1. Manuel Şifreleme

Manuel implementasyonlar, özellikle klasik şifreleme algoritmalarında kullanılmıştır. Bu sayede harf dönüşümleri, anahtar kullanımı ve şifreleme mantığı doğrudan kod seviyesinde gözlemlenmiştir. Manuel yöntemler algoritmaların eğitsel açıdan anlaşılmasını sağlamıştır.

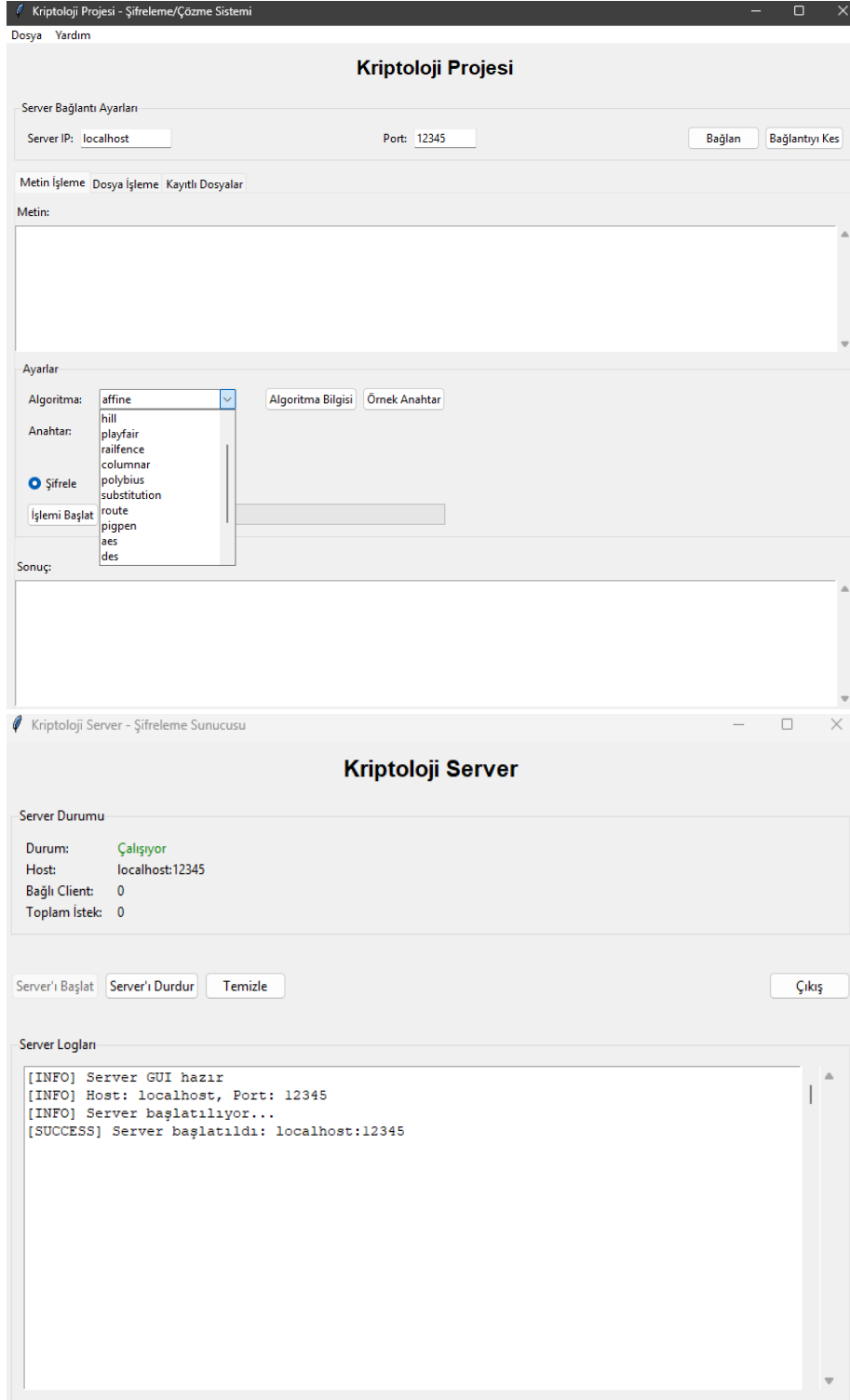
4.2. Kütüphane Tabanlı Şifreleme

AES, DES ve RSA algoritmaları Python kriptografi kütüphaneleri kullanılarak uygulanmıştır. Bu yöntem, güvenli, hızlı ve standartlara uygun sonuçlar üretmiştir.

4.3. Karşılaştırma

- Manuel şifreleme, öğrenme ve analiz açısından faydalıdır.
- Kütüphane tabanlı şifreleme, performans ve güvenlik açısından üstündür.

- Gerçek sistemlerde manuel şifreleme önerilmez



5. Wireshark ile Ağ Trafiği Analizi

Uygulama, Wireshark analizine uygun olacak şekilde tasarlanmıştır. Wireshark modu aktif edildiğinde, istemci ve sunucu arasındaki tüm TCP trafiği JSON formatında izlenebilmektedir.

5.1. Şifreli Verinin Ağ Üzerindeki Görünümü

Wireshark analizlerinde, şifreleme yapılmadan gönderilen verilerin açıkça okunabildiği; AES, DES veya RSA ile şifrelenen verilerin ise tamamen anlamsız karakterlerden oluştuğu gözlemlenmiştir. Bu durum, şifrelemenin veri gizliliğini sağladığını açıkça göstermektedir.

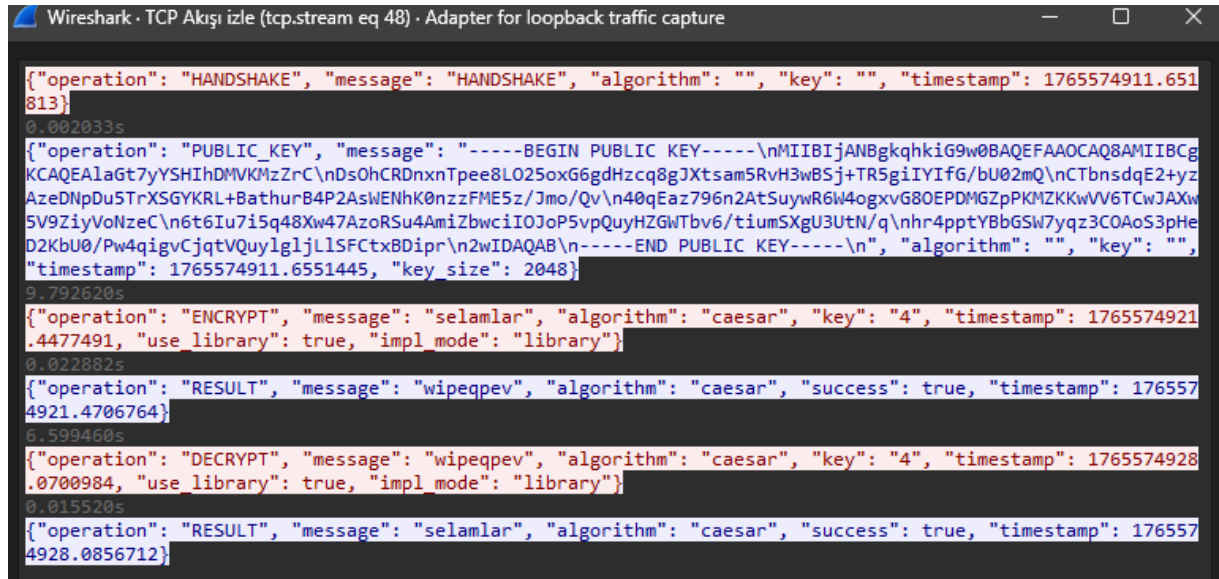
5.2. Paket Boyutu Analizi

- AES ve DES kullanıldığında paket boyutları daha küçüktür.
- RSA kullanıldığında, anahtar boyutu nedeniyle paketler belirgin şekilde büyümektedir.

Bu analiz, RSA'nın neden büyük veri şifrelemede değil, anahtar paylaşımında kullanıldığını doğrulamaktadır.

6. Manuel Şifreleme Yapılarına İlişkin Öğrenci Yorumu

Manuel ve klasik şifreleme algoritmaları, kriptografinin temelini anlamak açısından oldukça faydalıdır. Ancak bu algoritmalar modern saldırılara karşı yeterli güvenlik sağlamaz. Kısa anahtar uzunlukları ve basit matematiksel yapıları nedeniyle kolayca kırılabilirler. Bu nedenle gerçek dünyadaki uygulamalarda AES ve RSA gibi modern algoritmaların kullanılması zorunludur.



```
Wireshark · TCP Akışı izle (tcp.stream eq 48) · Adapter for loopback traffic capture

{"operation": "HANDSHAKE", "message": "HANDSHAKE", "algorithm": "", "key": "", "timestamp": 1765574911.651813}
0.002033s

{"operation": "PUBLIC_KEY", "message": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAlaGt7yYSHIhDMVKMzZrC\nDsOhCRDnXnTpee8L025oxG6gdHzcq8gJXtsam5RvH3wBSj+TR5giIYIfG/bU02mQ\nnCTbnsdqE2+yzAzeDnpDu5TrXSGYKRL+BathurB4P2AsWENhK0nzzFME5z/Jmo/Qv\nn40qEaz796n2AtSuywR6W4ogxvG80EPDMGZpPKMZKKwVV6TCwJAXw5V9ZiyVoNzeC\nn6t6Iu7i5q48Xw47AzoRSu4AmizbwciIOJoP5vpQuyHZGWTbv6/tiumSXgU3UtN/q\nnhr4pptYBbGSW7yqz3COAoS3pHeD2KbU0/Pw4qigvCjqtVQuylgljLLSFctxBDipr\nn2wIDAQAB\n-----END PUBLIC KEY-----\n", "algorithm": "", "key": "", "timestamp": 1765574911.6551445, "key_size": 2048}
9.792620s

{"operation": "ENCRYPT", "message": "selamlar", "algorithm": "caesar", "key": "4", "timestamp": 1765574921.4477491, "use_library": true, "impl_mode": "library"}
0.022882s

{"operation": "RESULT", "message": "wpeqpev", "algorithm": "caesar", "success": true, "timestamp": 1765574921.4706764}
6.599460s

{"operation": "DECRYPT", "message": "wpeqpev", "algorithm": "caesar", "key": "4", "timestamp": 1765574928.0700984, "use_library": true, "impl_mode": "library"}
0.015520s

{"operation": "RESULT", "message": "selamlar", "algorithm": "caesar", "success": true, "timestamp": 1765574928.0856712}
```