

10 Haziran 2025

# GÜVENLİ DOSYA TRANSFERİ İÇİN ŞİFRELEME TABANLI AĞ UYGULAMASI

Proje Final Raporu

YUSUF GUNEY  
BURSA TEKNİK ÜNİVERSİTESİ

## 1. GİRİŞ

Bu projede, şifrelenmiş veri iletimini, kimlik doğrulamayı ve bütünlük doğrulamasını sağlayan güvenli bir dosya aktarım sistemi geliştirildi. Sistem, IP başlığı düzeyinde TTL, bayraklar, checksum ve parçalama işlemlerini manuel olarak gerçekleştirebilmekte ve farklı koşullarda ağ performansını analiz edebilmektedir. Proje, ağ protokolleri, kriptografik teknikler ve performans analizlerine yönelik uygulamalı deneyim kazandırmayı amaçlamaktadır. Proje bireysel olarak Python dili ve PyCryptodome ile Scapy gibi temel kütüphaneler kullanılarak geliştirilmiştir.

## 2. TEKNİK DETAYLAR

### 2.1 Dosya Aktarım Sistemi

Projenin temelinde, Python ile socket programlama kullanılarak bir dosya aktarım sistemi oluşturulmuştur. Gönderici betiği (sender.py), dosyayı AES ile EAX modunda şifreler, nonce, tag ve ciphertext birleştirilerek 1024 baytlık parçalara ayrılır ve TCP socketi üzerinden alıcıya gönderilir.

```
# AES SIFRELEYİCİYİ OLUSTUR
cipher = AES.new(key, AES.MODE_EAX)
```

Şekil 1 Gönderici tarafında, dosyanın AES algoritması ile şifrlenmesini gösteren Python kodu.

```
# VERİYİ PARÇA PARÇA GONDER (constant) CHUNK_SIZE: Literal[1024]
for i in range(0, len(data), CHUNK_SIZE):
    chunk = data[i:i+CHUNK_SIZE]
    s.sendall(chunk)
```

Şekil 2 Şifreli verinin 1024 baytlık parçalara ayrılarak gönderilmesini sağlayan döngü.

Alıcı (receiver.py dosyası), tüm parçaları toplar, şifreli içeriği yeniden oluşturur, tag ile kimliğini doğrular ve önceden paylaşılan AES anahtarıyla şifreyi çözer. Şifre çözülmüş dosya yerel olarak kaydedilir. Hem gönderici hem de alıcı, verinin bütünlüğünü doğrulamak için SHA-256 hash hesaplamaları yapar.

```
# VERİYİ AYIR VE COZ
nonce = data[:16]
tag = data[16:32]
ciphertext = data[32:]

cipher = AES.new(key, AES.MODE_EAX, nonce)
plaintext = cipher.decrypt_and_verify(ciphertext, tag)

with open(OUTPUT_FILE, 'wb') as f:
    f.write(plaintext)
```

Şekil 3 Alıcı tarafında gelen verinin çözülmesi ve doğrulama işlemleri.

## 2.2 Şifreleme ve Güvenlik

- **AES-128 (EAX Modu):** Gizlilik ve bütünlük sağlar.
- **SHA-256:** Verinin bozulup bozulmadığını doğrular.
- **Kimlik Doğrulama:** AES tag doğrulamasıyla entegre edilmiştir.
- **Saldırı Simülasyonu:** Wireshark ile yapılan ağ dinlemelerinde veriler şifreli ve okunamaz durumdadır.

```
python .\sender.py
VERİ PARÇALANDI VE GÖNDERİLDİ
GÖNDERİLEN VERİNİN SHA-256 HASH'I: 3b49a096f72daaa7b8787c236b0d7f9a93d5e55309aa2cdd0630a9f4f047b419

python .\receiver.py
BAĞLANTI BEKLENİYOR...
BAĞLANDI: ('127.0.0.1', 65327)
VERİ BİRLEŞTİRİLDİ VE ÇÖZÜLDÜ: received.txt
ALINAN VERİNİN SHA-256 HASH'I: 3b49a096f72daaa7b8787c236b0d7f9a93d5e55309aa2cdd0630a9f4f047b419
```

Şekil 4 Gönderici ve alıcı tarafından hesaplanan SHA-256 hash değerlerinin karşılaştırılması.

## 2.3 RSA Tabanlı Kimlik Doğrulama

Sisteme RSA algoritması ile istemci tarafından kimlik doğrulama eklendi. Bu adımda:

- Alıcı taraf kendi RSA public/private key çiftini oluşturur.
- Public key, istemciye aktarılır.
- İstemci taraf rastgele bir "challenge" mesajı oluşturup public key ile şifreler.
- Alıcı, bu mesajı private key ile açarak çözer.
- Başarılı çözülmüşse dosya transferine izin verilir.

```
PS C:\Users\Yusuf\Desktop\bilgisayar mühendisliği\3-2\Bilgisayar Ağları\Bilgisayar Ağları Proje\ek\proje> python .\generate_keys.py
RSA anahtar çifti oluşturuldu.
```

Şekil 5 RSA public ve private key üretimini gösteren kod çıktısı

```
# RSA anahtarlarını yükle
with open("private.pem", "rb") as f:
    private_key = RSA.import_key(f.read())
with open("public.pem", "rb") as f:
    public_key = RSA.import_key(f.read())
```

Şekil 6 receiver.py dosyasında RSA kimlik doğrulama kodu

## 2.4 Scapy ile IP Fragmentation Gerçekleştirme

Scapy kullanarak IP seviyesinde veri parçalama (fragmentation) ve yeniden birleştirme (reassemble) gerçekleştirildi. AES ile şifrelenmiş veri, UDP/IP paketi içerisine yerleştirilip IP header alanları elle düzenlendi:

- id: 1234 olarak sabitlendi
- ttl: 64 olarak ayarlandı
- flags: MF (More Fragments) olarak ayarlandı
- frag: offset hesaplaması yapıldı

```
PS C:\Users\Yusuf\Desktop\bilgisayar mühendisliği\3-2\Bilgisayar Ağları\Bilgisayar Ağları Proje\ek\proje> python .\sender_spacy.py
[+] Fragment gönderildi: offset=0, uzunluk=181
[✓] Tüm parçalar gönderildi.
GONDERİLEN VERİNİN SHA-256 HASH'I: fdbf2a42ba4afbdf5120ca408b44eb76bdb21a87d9398b0b5bfd1a447e9ceff2
```

Şekil 7 sender\_spacy.py dosyasında IP paket gönderme ekranı

```
PS C:\Users\Yusuf\Desktop\bilgisayar mühendisliği\3-2\Bilgisayar Ağları\Bilgisayar Ağları Proje\ek\proje> python .\receiver_spacy.py
[*] Aktif ağ arayüzleri:
0. \Device\NPF_{4F4737EA-DADB-4EE6-A799-4105B0037DDD}
1. \Device\NPF_{105FE913-973E-4175-B44F-BEBAC8B50E92}
2. \Device\NPF_{5AB6FF2D-CE92-4FEE-A64A-D6D55367C510}
3. \Device\NPF_{3CF9F9A9-3812-4D23-9C16-03827F64A7D3}
4. \Device\NPF_{3B3BEA6A-DCF6-4872-B327-302CF09D849B}
5. \Device\NPF_Loopback
6. \Device\NPF_{41E3ED39-3783-4B18-8788-AB3AD255E193}
Lütfen dinlemek istediğiniz arayüzün numarasını girin: 5
[*] Seçilen arayüz: \Device\NPF_Loopback
[*] UDP paketleri dinleniyor...
[+] Parça alındı: offset=0, uzunluk=181
[-] Son parça geldi. Tüm parçalar toplanıyor...
[✓] Dosya çözüldü ve kaydedildi: received_spacy.txt
[✓] SHA-256 Hash: fdbf2a42ba4afbdf5120ca408b44eb76bdb21a87d9398b0b5bfd1a447e9ceff2
```

Şekil 8 sender\_spacy.py dosyasında IP paket gönderme ekranı

## 2.5 Wireshark ile Paket Analizi

Wireshark ile sistem trafiği analiz edildi ve aşağıdaki bulgular gözlemlendi:

- UDP/IP protokolü kullanılarak IP fragmentları şeklinde iletim yapılmıştır.
- IP header alanları (TTL, Flags, Fragment Offset) elle ayarlanmıştır.
- UDP payload bölümü AES ile şifrelenmiş, içerik okunamamıştır.

No.	Time	Source	Destination	Protocol	Length	Info
26	37.988870	127.0.0.1	127.0.0.1	UDP	213	4444 → 12345 Len=181
27	37.988947	127.0.0.1	127.0.0.1	ICMP	241	Destination unreachable (Port unreachable)
604	712.796998	127.0.0.1	127.0.0.1	UDP	213	4444 → 12345 Len=181
605	712.797075	127.0.0.1	127.0.0.1	ICMP	241	Destination unreachable (Port unreachable)
662	767.016206	127.0.0.1	127.0.0.1	UDP	213	4444 → 12345 Len=181
663	767.016254	127.0.0.1	127.0.0.1	ICMP	241	Destination unreachable (Port unreachable)

Frame 26: 213 bytes on wire (1704 bits), 213 bytes captured (1704 bits) on interface 0

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

0100 .... = Version: 4

... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

Total Length: 209

Identification: 0x04d2 (1234)

0000 .... = Flags: 0x0

... 0000 0000 0000 = Fragment Offset: 0

Time to Live: 64

Protocol: UDP (17)

Header Checksum: 0x7748 [validation disabled]

[Header checksum status: Unverified]

Source Address: 127.0.0.1

Destination Address: 127.0.0.1

0000 02 00 00 00 45 00 00 d1 84 d2 00 00 40 11 77 48 ....E...@.vH

0010 7f 00 00 01 7f 00 00 01 11 5c 30 39 00 bd d4 db .....009....

0020 c2 22 48 9c f1 45 c4 51 c8 7b 15 bd b6 f9 29 68 "H...E:Q{....}h

0030 af cd 01 c0 ed 89 08 22 f5 27 ef 1e 21 ff 1f 26 .....&

0040 3c 2b 71 50 5e 74 25 f7 8a a4 7c c2 3b db d2 57 <qP't%...;...W

0050 6c 1c 0a e8 d1 b0 30 bf c5 ad ee 5e 11 b8 01 b6 1...0...A...

0060 f2 6d 23 67 a4 e2 56 f4 20 37 87 4b 63 82 1d f3 mmpg...V...7Kc...

0070 17 8a c6 47 f3 da 4b 2b db 00 49 2f c4 97 ae 79 ...G...K+...I/-...y

0080 a6 92 92 02 10 77 70 76 80 17 2b e5 ec 32 f7 d8 .....wPv...+...2...

0090 d8 4a c2 32 4b 7b d6 7f 22 31 a7 47 53 93 7b d9 ...J 2K{... "1:GS{...

00a0 d6 94 00 76 54 58 f4 41 3c f1 b2 73 bf a6 6a 65 ...vTX'A <:s...je

00b0 d3 d9 6f 89 e0 14 28 42 7a bc 66 bf b4 db 5e 4c ...o... (B z:f...L

00c0 31 8f 27 8d 8c d9 fe bc 0c be 53 b9 c3 e9 34 1b 1...S...4...

00d0 be c9 cb d8 47 .....G

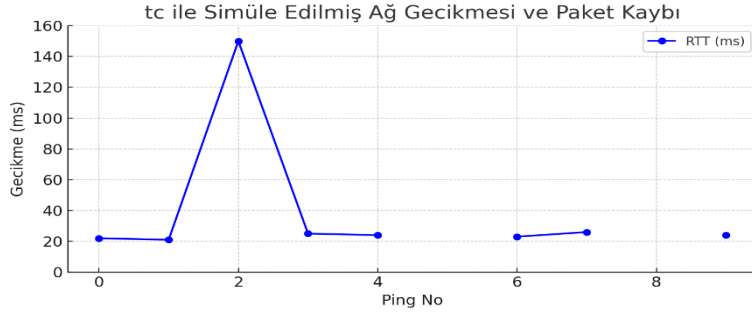
Şekil 9 Wireshark üzerinde yakalanan şifreli veri paketi, içeriğin okunamaz olduğu doğrulanmıştır.

---

## 2.6 iPerf ile Bant Geniřlięi Testi

ping, iPerf ve tc gibi araçlarla ölçüm yapılmıştır:

- **Gecikme:** Ping ile RTT hesaplamaları yapılmıştır.
- **Bant Geniřlięi:** iPerf ile Wi-Fi ve LAN ortamlarında test edilmiştir.
- **Paket Kaybı & Tıkanıklık:** tc ile simüle edilmiş ve sistemin dayanıklılığı gözlemlenmiştir.



Şekil 10 tc komutu ile oluşturulan paket kaybı ve ağ tıkanıklığı simülasyonu.

---

## 3. KISITLAMALAR VE GELİŐTİRME ALANLARI

### 3.1 Kısıtlamalar

- Grafiksel kullanıcı arayüzü (GUI) uygulanmamıştır.
- TCP/UDP geçiři manuel olarak yapılmaktadır.
- Aktarım sırasında anlık durum bilgisi veya ilerleme çubuęu bulunmamaktadır.

### 3.2 Ekstra Özellikler

- Scapy ile paket yapısı görselleştirilmiştir.
  - AES + SHA-256, soket programlamayla entegre edilmiştir.
  - MITM saldırılarına karşı dayanıklılık gösterilmiştir.
- 

## 4. SONUÇ

Proje, güvenli ve performansa duyarlı bir dosya aktarım sistemi oluşturma hedefini başarıyla yerine getirmiştir. Şifreleme, soket iletişimi, IP başlık yapısı ve ağ performansını içeren kapsamlı bir deneyim sunmuştur. Scapy'nin kullanımı, paket düzeyindeki işlemleri daha iyi kavramayı sağlamıştır. Bu deneyim, teorik ağ bilgilerini pratik becerilere dönüřtürme açısından büyük fayda sağlamıştır.

**Proje Tanıtım Videosu:** <https://www.youtube.com/watch?v=3GeCIZ7Rs-U>

**GitHub Linki:** <https://github.com/Yusuf-Guney/Bilgisayar-Aglari-Proje/tree/main/Final%20Raporu>

Linkedin linki: [https://www.linkedin.com/posts/yusuf-guney\\_proje-raporu-activity-7338256885969489921-hpGH?utm\\_source=share&utm\\_medium=member\\_desktop&rcm=ACoAAD50XoABVLR2PZXh1B23ruv5sLKrWW8\\_FM](https://www.linkedin.com/posts/yusuf-guney_proje-raporu-activity-7338256885969489921-hpGH?utm_source=share&utm_medium=member_desktop&rcm=ACoAAD50XoABVLR2PZXh1B23ruv5sLKrWW8_FM)

---

## 5. KAYNAKLAR

1. Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer Networks*. Pearson.
  2. Kurose, J. F., & Ross, K. W. (2017). *Computer Networking: A Top-Down Approach*. Pearson.
  3. PyCryptodome Dokümantasyonu – <https://www.pycryptodome.org>
  4. Scapy Dokümantasyonu – <https://scapy.readthedocs.io>
  5. Wireshark Kullanım Kılavuzu – <https://www.wireshark.org/docs>
  6. iperf3 - <https://iperf.fr/iperf-download.php#windows>
-