

Ahmet Yusuf Birdir

21360859026

Bilgisayar Mühendisliği 3. Sınıf Öğrencisi

Dosya Transfer Sistemi

GİRİŞ

Bu proje kapsamında, güvenli veri iletimi ve ağ performansı analizine odaklanan bir Gelişmiş Dosya Aktarım Sistemi geliştirilmiştir. Sistem, dosya transferi sırasında gizlilik, bütünlük ve kimlik doğrulama gibi temel güvenlik gereksinimlerini sağlarken, aynı zamanda manuel parçalara ayırma ve yeniden birleştirme gibi işlemleri gerçekleştirebilecek şekilde tasarlanmıştır.

Proje; kriptografik protokoller (AES/RSA), paket seviyesinde veri bütünlüğü denetimi (SHA-256), istemci-sunucu kimlik doğrulaması, dinamik tıkanıklık kontrolü ve UDP üzerinden hata toleranslı veri iletimi gibi çeşitli güvenlik ve ağ iletişimi mekanizmalarını bütünlük olarak içermektedir. Ayrıca, proje süresince Wireshark ve iPerf3 gibi araçlar kullanılarak ağ performansı değerlendirilmiş, farklı senaryolar altında veri iletim süreleri, paket kayıpları ve bant genişliği kullanımı analiz edilmiştir.

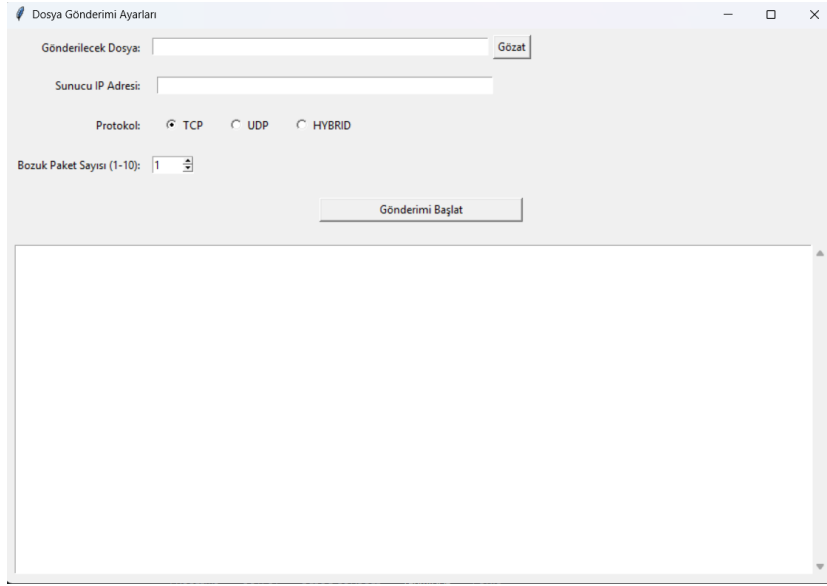
Bu rapor, geliştirilen sistemin teknik mimarisini, uygulanan güvenlik önlemlerini, karşılaşılan zorlukları, performans analizlerini ve iyileştirme önerilerini detaylı biçimde sunmaktadır.

Teknik Detaylar

2.1 Sistem Mimarisi

Sistem istemci-sunucu modeline dayalı olarak yapılandırılmıştır. İstemci tarafında kullanıcı arayüzü aracılığıyla dosya seçimi ve iletim parametreleri ayarlanırken, sunucu tarafında gelen bağlantılar

dinlenmekte ve doğrulananan istemcilerden şifrelenmiş dosya parçaları alınarak bütünleştirme işlemi yapılmaktadır. Sistem Şekil 1 ve Şekil 2’ de görülmektedir.



Şekil 1 – Client Arayüz İçeriği

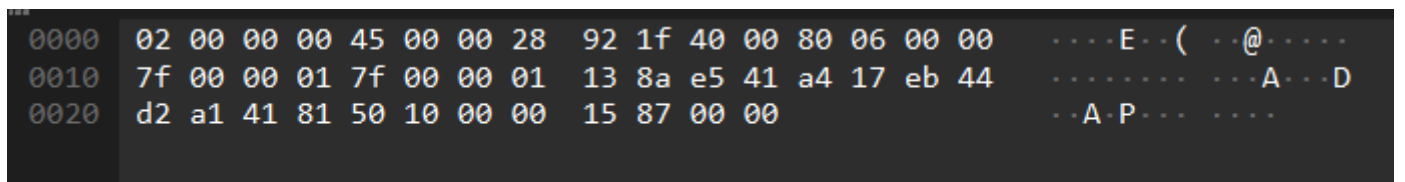
```
Starting iperf3 server on port 5001...
Starting iperf3 server on port 5002...
iperf3 servers are running on ports 5001 and 5002.
[TCP] Dinleniyor: 0.0.0.0:5001
[*] Server başlatıldı. TCP ve UDP dinleniyor...
[UDP] Dinleniyor: 0.0.0.0:5002
[TCP] Bağlantı: ('127.0.0.1', 54848)
[TCP] Parça alındı: 1/396
[TCP] Parça alındı: 2/396
[TCP] Parça alındı: 3/396
[TCP] Parça alındı: 4/396
[TCP] Parça alındı: 5/396
```

Şekil 2 – Server Çıktısı

2.2 Dosya Parçalama ve Şifreleme Mekanizması

Gönderilecek dosya, önceden belirlenen sabit boyutlarda parçalara ayrılır. Her parça, AES algoritması ile simetrik olarak şifrelenir. Simetrik anahtar ise RSA algoritması kullanılarak alıcının açık anahtarıyla şifrelenir. Bu yöntem sayesinde hem veri gizliliği hem de güvenli anahtar iletimi sağlanmaktadır.

Her şifrelenmiş parçanın SHA-256 özeti hesaplanarak paketin içerisine eklenir. Böylece alıcı tarafında veri bütünlüğü doğrulanabilir hale gelir.



0000	02 00 00 00 45 00 00 28 92 1f 40 00 80 06 00 00	...E..(..@.....
0010	7f 00 00 01 7f 00 00 01 13 8a e5 41 a4 17 eb 44A..D
0020	d2 a1 41 81 50 10 00 00 15 87 00 00	..A.P.....

Şekil 3– Wireshark Analizi

2.3 Kimlik Doğrulama

İstemci, sunucuya bağlantı kurmadan önce sabit bir kimlik doğrulama belirteci (token) göndererek sisteme erişim yetkisini ispatlar. Bu belirteç her iki uçta da sabit olup, yetkisiz erişimleri engellemek amacıyla kullanılır.

2.4 TCP ile Dosya Aktarımı

TCP üzerinden iletim sırasında istemci, dosya parçalarını sırasıyla sunucuya gönderir. Her paketin başında sıra numarası, uzunluk bilgisi ve SHA-256 özeti yer alır. Sunucu tarafında bu özet doğrulanarak veri bütünlüğü sağlanır, ardından parça AES ile çözülüp disk üzerine yazılır.

2.5 UDP ile Dosya Aktarımı ve Hata Yönetimi

UDP üzerinden dosya aktarımında, güvenilirlik protokol düzeyinde garanti edilmediği için istemci tarafında özel bir **ACK/NACK tabanlı hata yönetimi** uygulanmaktadır. Her gönderilen paketten sonra sunucudan ACK (kabul) ya da NACK (red) yanıtı beklenir. NACK alınması veya zaman aşımı yaşanması durumunda paket tekrar gönderilir.

```
[iperf3] 5001 portuna bağlanıyor...
[iperf3] 5002 portuna bağlanıyor...
Bozuk paket sayısı: 5
[BOZUK] Paket 30 bozuldu.
[BOZUK] Paket 114 bozuldu.
[BOZUK] Paket 232 bozuldu.
[BOZUK] Paket 262 bozuldu.
[BOZUK] Paket 356 bozuldu.
[UDP] Dosya gönderimi başlıyor: deneme.pdf
```

Şekil 4- Paket Bozma

```
[Adaptif] Gecikme düşük (0.02s), hız artırıldı. delay = 0.005s
[UDP] ACK alındı: 261
[Adaptif] Gecikme düşük (0.03s), hız artırıldı. delay = 0.005s
[UDP] NACK alındı, paket 262 tekrar gönderiliyor.
[UDP] ACK alındı: 262
[Adaptif] Gecikme düşük (0.06s), hız artırıldı. delay = 0.005s
[UDP] ACK alındı: 263
[Adaptif] Gecikme düşük (0.03s), hız artırıldı. delay = 0.005s
[UDP] ACK alındı: 264
[Adaptif] Gecikme düşük (0.04s), hız artırıldı. delay = 0.005s
```

2.6

Şekil 5- Paket Yeniden Gönderimi ve Hız Ayarı

HYBRID Dosya Aktarımı

Hybrid protokol seçiminde, öncelikle hedef IP adresine ping testi gerçekleştirilir ve ağ gecikme süresi ölçülür. Elde edilen ortalama gecikme belirlenen eşik değerine göre değerlendirilir. Gecikme düşükse (örneğin 100 ms altında) hızlı ve düşük gecikmeli iletişim sağlamak amacıyla UDP protokolü tercih edilir. Ancak UDP, paket kayıplarını doğrudan garanti etmediğinden, istemci tarafında hata yönetimi için ACK/NACK mekanizması kullanılır; paket onayı alınmazsa veya zaman aşımı gerçekleşirse paket yeniden gönderilir. Gecikme yüksek olduğunda ise, daha güvenilir ve bağlantı odaklı TCP protokolü seçilir. Bu yöntemle hem ağ koşullarına uygun en iyi protokol dinamik olarak belirlenmiş olur hem de aktarımın güvenilirliği sağlanır.

```
HYBRID protocol seçildi.
Server'a ping atılıyor.
Ping testi çıktısı:

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

Ortalama gecikme (ms): 0
delay değeri 100ms'in altında olduğundan udp protokolü seçildi.
[iperf3] 5002 portuna bağlanıyor...
[iperf3] 5001 portuna bağlanıyor...
Bozuk paket sayısı: 3
[BOZUK] Paket 6 bozuldu.
[BOZUK] Paket 14 bozuldu.
```

Şekil 6- Paket Yeniden Gönderimi ve Hız Ayarı

2.7 Dinamik Tıkanıklık Kontrolü

UDP modunda veri iletiminde, her paketin RTT süresi ölçülerek gönderim aralığı dinamik olarak ayarlanır. Gecikme yüksekse gönderim yavaşlatılır, düşükse hız artırılır. Bu sayede ağ bant genişliği verimli kullanılmakta ve tıkanıklık önlenmektedir. Şekil 5 'te aralık ayarlama işlemi görülmektedir.

2.8 Wifi vs Ethernet – Remote vs Local

Burada iperf3 ile bağlantı sırasındaki bant genişliği ölçülmüş ve analiz edilmiştir.

```
[iperf3] Test sonuçları iç:
[iperf3 port 5002]:
Connecting to host 127.0.0.1, port 5002
[ 4] local 127.0.0.1 port 54851 connected to 127.0.0.1 port 5002
[ ID] Interval            Transfer      Bandwidth
[ 4]  0.00-1.01      sec    194 MBytes  1.62 Gbits/sec
[ 4]  1.01-2.01      sec    207 MBytes  1.73 Gbits/sec
[ 4]  2.01-3.00      sec    198 MBytes  1.68 Gbits/sec
[ 4]  3.00-4.01      sec    218 MBytes  1.82 Gbits/sec
[ 4]  4.01-5.00      sec    226 MBytes  1.90 Gbits/sec
[ 4]  5.00-6.01      sec    234 MBytes  1.95 Gbits/sec
[ 4]  6.01-7.00      sec    236 MBytes  1.99 Gbits/sec
[ 4]  7.00-8.01      sec    227 MBytes  1.89 Gbits/sec
[ 4]  8.01-9.01      sec    215 MBytes  1.79 Gbits/sec
[ 4]  9.01-10.00     sec    190 MBytes  1.61 Gbits/sec
-----
[ ID] Interval            Transfer      Bandwidth
[ 4]  0.00-10.00     sec    2.09 GBytes  1.80 Gbits/sec
[ 4]  0.00-10.00     sec    2.09 GBytes  1.80 Gbits/sec
sender
receiver

iperf Done.
```

Şekil 7 Local - TCP - Ethernet

```
[iperf3] Test sonuçları iç:
[iperf3 port 5002]:
Connecting to host 192.168.1.113, port 5002
[ 4] local 192.168.1.104 port 54587 connected to 192.168.1.113 port 5002
[ ID] Interval            Transfer      Bandwidth
[ 4]  0.00-1.00      sec    66.6 MBytes  559 Mbits/sec
[ 4]  1.00-2.00      sec    50.6 MBytes  424 Mbits/sec
[ 4]  2.00-3.00      sec    71.2 MBytes  597 Mbits/sec
[ 4]  3.00-4.00      sec    81.1 MBytes  680 Mbits/sec
[ 4]  4.00-5.01      sec    75.2 MBytes  628 Mbits/sec
[ 4]  5.01-6.00      sec    64.5 MBytes  544 Mbits/sec
[ 4]  6.00-7.01      sec    70.8 MBytes  589 Mbits/sec
[ 4]  7.01-8.00      sec    63.0 MBytes  531 Mbits/sec
[ 4]  8.00-9.01      sec    64.1 MBytes  534 Mbits/sec
[ 4]  9.01-10.00     sec    4.86 MBytes  41.3 Mbits/sec
-----
[ ID] Interval            Transfer      Bandwidth
[ 4]  0.00-10.00     sec    612 MBytes  513 Mbits/sec
[ 4]  0.00-10.00     sec    612 MBytes  513 Mbits/sec
sender
receiver

iperf Done.
```

Şekil 8 Remote - TCP - Ethernet


```
[iperf3] Test sonuçları iç:
[iperf3 port 5002]:
Connecting to host 192.168.1.113, port 5002
[ 4] local 192.168.1.106 port 54808 connected to 192.168.1.113 port 5002
[ ID] Interval          Transfer      Bandwidth
[ 4] 0.00-1.01 sec      5.04 MBytes  41.8 Mbits/sec
[ 4] 1.01-2.00 sec      3.38 MBytes  28.7 Mbits/sec
[ 4] 2.00-3.00 sec      3.01 MBytes  25.2 Mbits/sec
[ 4] 3.00-4.00 sec      3.26 MBytes  27.3 Mbits/sec
[ 4] 4.00-5.01 sec      4.98 MBytes  41.7 Mbits/sec
[ 4] 5.01-6.00 sec      4.80 MBytes  40.5 Mbits/sec
[ 4] 6.00-7.00 sec      5.66 MBytes  47.3 Mbits/sec
[ 4] 7.00-8.00 sec      6.15 MBytes  51.8 Mbits/sec
[ 4] 8.00-9.00 sec      5.35 MBytes  44.9 Mbits/sec
[ 4] 9.00-10.01 sec     5.66 MBytes  47.3 Mbits/sec
- - - - -
[ ID] Interval          Transfer      Bandwidth
[ 4] 0.00-10.01 sec     47.3 MBytes  39.7 Mbits/sec
[ 4] 0.00-10.01 sec     47.3 MBytes  39.7 Mbits/sec
sender
receiver

iperf Done.
```

Şekil 9 Remote - TCP - Wifi

```
Ping testi başlatılıyor: 192.168.1.113
Ping sonucu:

Pinging 192.168.1.113 with 32 bytes of data:
Reply from 192.168.1.113: bytes=32 time=12ms TTL=128
Reply from 192.168.1.113: bytes=32 time=25ms TTL=128
Reply from 192.168.1.113: bytes=32 time=11ms TTL=128
Reply from 192.168.1.113: bytes=32 time=9ms TTL=128

Ping statistics for 192.168.1.113:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 25ms, Average = 14ms
```

Şekil 10 Ping testi

2.9 KOD ANLATIMI

Bu bölümde, istemci ve sunucu uygulamalarında kullanılan Python kodlarının teknik açıklamaları detaylı olarak sunulmuştur. Kod parçaları, şifreleme, protokol seçimi, paket bozma, dinamik hız ayarı, dosya gönderimi ve alınması gibi temel mekanizmaları örneklerle göstermektedir.

2.9.1 client.py Fonksiyonları

1. gui_log(message)

Bu fonksiyon, bir mesajı GUI'deki log ekranına yazmak için kullanılır. Tkinter GUI thread'i dışında doğrudan widget'a erişim hataya yol açabileceğinden, root.after(0, ...) ile GUI thread'inde güvenli biçimde çalışacak şekilde mesaj ekler. Log ekranı önce düzenlenebilir hâle getirilir, mesaj eklenir, sonra tekrar kilitlenir.

2. measure_ping(ip)

Belirtilen IP adresine 4 adet ping paketi gönderir ve ortalama gecikmeyi milisaniye cinsinden döndürür. İşletim sistemine göre (Windows veya Linux/macOS) farklı parametreler kullanır ve subprocess.check_output ile komut çıktısını okur. Regex kullanarak ortalama gecikmeyi çıkarır ve GUI'ye loglar.

3. run_iperf3_client(server_ip, port, gui_log)

Verilen server_ip ve port için iperf3 aracını çalıştırır ve ağ performansı (bant genişliği) ölçümü yapar. subprocess ile başlatılan komutun çıktısı satır satır okunur, sonuçlar iperf3_results listesine eklenir ve GUI'ye yazdırılır.

4. start_transfer_thread()

Dosya gönderimini GUI'yi dondurmadan başlatmak için transfer_file() fonksiyonunu yeni bir thread içinde çalıştırır. Bu sayede GUI arayüzü açık kalırken dosya aktarımı arka planda devam eder.

5. corrupt_packets(data_packets, corrupt_count)

Verilen veri paketlerinden rastgele corrupt_count kadarını bozar. Her bozulan pakette ilk bayt ters çevrilir (XOR işlemi uygulanır). Bozuk paketler log'a yazılır. Geriye bozulmuş veya bozulmamış paketlerin listesini döner.

6. send_packet_with_retries(seq, first_packet, original_packet, server_addr, sock, delay)

UDP üzerinden bir veri paketi gönderir. İlk gönderimde bozulmuş paket kullanılır, sonra orijinal paket tekrar gönderilir. Sunucudan ACK gelmezse NACK veya timeout durumunda 10 kez tekrar denenir. ACK gelirse başarıyla tamamlanır, aksi takdirde hata fırlatılır. Gönderilen paketin sırası, hash değeri ve veri kısmı yapılandırılarak gönderilir.

7. transfer_file()

Dosya transfer sürecinin merkezidir. Şu işlemleri yapar:

1. Kullanıcının protokol seçimine göre TCP, UDP veya HYBRID belirler.
2. iperf3 ile performans testi başlatır.
3. Dosyayı okur, 1024 byte'lık parçalara böler.
4. Her parçayı AES ile şifreler, SHA-256 ile hash'ler.
5. AES anahtarını RSA ile şifreleyip sunucuya yollar.
6. Belirli sayıda paketi bozar (corrupt_packets).
7. **UDP seçilirse:**
 - o Dosya adı ve AES anahtarı UDP ile gönderilir.

- Her paket `send_packet_with_retries()` ile gönderilir.
- RTT'ye göre gönderim hızı ayarlanır.

8. TCP seçilirse:

- Kimlik doğrulama için token gönderilir.
- Dosya adı, şifreli anahtar, toplam parça sayısı gönderilir.
- Her parça başlığı ve içeriği sıralı biçimde TCP ile gönderilir.

9. Süreç boyunca loglama yapılır, hata varsa kullanıcıya bildirilir.

8. `open_gui()`

Tkinter GUI arayüzünü oluşturur. Kullanıcının şu bilgileri girmesini sağlar:

- Gönderilecek dosya (dosya seçici ile)
- Sunucu IP adresi
- Kullanılacak protokol (TCP, UDP, HYBRID)
- Bozuk paket sayısı (0–10 arası)

Ayrıca bir log ekranı içerir. “Gönderimi Başlat” butonuna basıldığında girdiler doğrulanır, geçerliyse `start_transfer_thread()` çağrılır ve transfer başlatılır.

2.9.2 `server.py` Fonksiyonları

1. `recv_exact(conn, size)`

TCP bağlantısı üzerinden, tam olarak istenilen miktarda (`size` byte) veri alınmasını sağlar. `recv()` çağrısı gelen verinin tamamını bir seferde döndürebilir; bu nedenle alınan veri `size` uzunluğa ulaşana kadar döngüyle veri toplanır.

2. `start_iperf3_servers()`

Port 5001 ve 5002 üzerinde iperf3 sunucularını arka planda başlatır. Bu sunucular istemcilerden gelen iperf3 bağlantılarını kabul eder, böylece ağ performansı (örneğin bant genişliği) ölçülebilir. `subprocess.Popen` ile sessiz şekilde çalıştırılır.

3. `stop_iperf3_servers()`

Sunucu kapanırken iperf3 süreçlerini düzgün biçimde sonlandırmak için çağrılır. Her iki iperf3 sunucusuna da SIGINT sinyali gönderilir ve kapanmaları beklenir. Bu işlem `atexit.register(...)` ile program kapanışına bağlanmıştır.

4. `tcp_server()`

TCP bağlantısı üzerinden istemciden gelen şifreli dosyayı güvenli bir şekilde alır ve diske yazar. Aşamalar:

1. TCP bağlantısı başlatılır ve dinlenir.
2. Kimlik doğrulama token'ı kontrol edilir.
3. Dosya adı alınır ve kayıt için dosya hazırlanır.
4. RSA ile şifrelenmiş AES anahtarı alınır ve özel anahtarla çözülür.
5. Toplam parça sayısı alınır.
6. Her parçanın başlığı (sıra, uzunluk, hash) ve şifreli içeriği alınır.

7. SHA-256 ile hash doğrulaması yapılır.
8. AES-CBC ile veri çözülür ve diske yazılır.
9. İşlem sonunda dosyanın alındığı loglanır.

5. udp_server()

UDP üzerinden gelen veri paketlerini güvenli bir şekilde alır, doğrular ve diske yazar. Çünkü UDP doğal olarak güvenilir değildir; bu nedenle aşağıdaki işlemler uygulanır:

1. FILENAME: ile dosya adı alınır, dosya hazırlanır.
2. KEY: ile AES anahtarı alınır.
3. CHUNK: paketlerinde:
 - Sıra numarası, hash ve veri ayrıştırılır.
 - Hash kontrolü yapılır.
 - Uygunsa ACK, hatalıysa NACK gönderilir.
 - Alınan paket hafızaya yazılır.
4. END: mesajıyla toplam parça sayısı alınır.
5. Bütün parçalar tamamlandıysa AES ile çözülüp sırayla diske yazılır.
6. İşlem bitince her şey sıfırlanır (bir sonraki dosya için temizlenir).

6. if __name__ == "__main__":

Program başlatıldığında sunucunun tüm işlevlerini aktif hale getirir:

- iperf3 sunucularını başlatır.
- tcp_server() ve udp_server() fonksiyonlarını ayrı thread'lerde başlatır.
- Sonsuz döngüyle çalışmayı sürdürür.
- CTR+C ile kesilirse iperf3 sunucuları kapatılır.

SINIRLAMALAR VE GELİŞTİRME ÖNERİLERİ

Bu bölümde geliştirilen sistemin mevcut sınırları ve ileriye dönük iyileştirme olanakları ele alınmaktadır. Uygulama kapsamı gereği bazı teknik bileşenler temel düzeyde bırakılmış, bazı gelişmiş özellikler ise zaman veya kaynak kısıtları nedeniyle sınırlı biçimde uygulanmıştır.

3.1 Uygulanan Ancak Sınırlı Kalan Özellikler

3.1.1 Dinamik Tıkanıklık Kontrolü

UDP aktarımında RTT bazlı basit bir gönderim gecikmesi ayarlama (delay adaptation) sistemi uygulanmıştır. Ancak bu yapı, modern ağ protokollerinde kullanılan gelişmiş algoritmalar kadar detaylı değildir ve projenin tcp protokolü kısmına uygulanmamıştır.

3.1.2 Grafiksel Performans Analizi

iPerf3 ve Wireshark verileri üzerinden analiz yapılmış ve okumakta olduğunuz raporun içerisinde gerekli kısımlara koyulmuş çıktı örnekleri bulunmaktadır. Ancak sonuçların kapsamı yüzeysel kalmış ve daha kapsamlı grafik analizlerine girilmemiştir.

3.2 Uygulanamayan Özellikler

3.2.1 Gerçek Zamanlı Saldırı Tespiti ve Filtreleme

MITM saldırısı simüle edilmiş ancak başarılı olmamıştır. Başarılı gerçekleştirilemeyen bu işlem proje kapsamından çıkarılmıştır.

3.1.2 Düşük Seviyeli IP Başlık İşleme

IP başlık alanları (TTL, checksum, DF/MF bayrakları) üzerinde doğrudan işlem yapılmamıştır.

3.3 Gelecekteki Geliştirme Önerileri

- **Tamamen Manuel IP Parçalama:** Özellikle MTU altı boyutlarda paketlere ayırıp, yeniden birleştirme işlemini tamamen elle yönetme.
- **Gerçek Zamanlı IDS Entegrasyonu:** MITM gibi saldırıların davranışsal olarak algılanıp otomatik engellenmesi.
- **Performans Verilerinin Görselleştirilmesi:** Matplotlib gibi kütüphaneler kullanılarak RTT, bant genişliği, hata oranı gibi metriklerin grafiksel olarak raporlanması.
- **Çoklu İstemci Desteği:** Şu anki yapı yalnızca tek istemciyi desteklemekte; eş zamanlı çoklu istemci desteğiyle sistemin ölçeklenebilirliği artırılabilir.

SONUÇ

Bu proje kapsamında geliştirilen **Gelişmiş Güvenli Dosya Aktarım Sistemi**, hem ağ iletişimi hem de bilgi güvenliği alanında ileri seviye teknik beceriler gerektiren birçok unsuru bir araya getirmiştir. Sistem, dosya transferlerini güvenli, bütünlüklü ve adaptif bir biçimde gerçekleştirmek üzere yapılandırılmıştır.

Aktarılan dosyaların gizliliği, AES simetrik şifreleme ile sağlanmış; bu anahtar ise RSA algoritması kullanılarak güvenli bir şekilde iletilmiştir. Paket bütünlüğü, her veri parçası için hesaplanan SHA-256 özeti ile doğrulanmıştır. Ayrıca istemci-sunucu doğrulaması, özel bir kimlik belirteci (token) aracılığıyla gerçekleştirilerek sisteme yetkisiz erişim engellenmiştir.

Dosya transferi sırasında, **TCP** protokolünün sunduğu güvenilirlikten yararlanıldığı gibi, **UDP** protokolüyle sağlanan hız avantajı da proje kapsamında ele alınmıştır. UDP iletiminde, hatalı paketler için NACK/ACK tabanlı özel bir yeniden iletim mekanizması geliştirilmiş ve **dinamik tıkanıklık kontrolü** ile ağ koşullarına göre iletim hızı ayarlanmıştır.

Ek olarak, sistemde uygulanan **hibrit protokol seçimi** sayesinde, ağ gecikmesi ve bant genişliği ölçümleri doğrultusunda TCP ve UDP protokolleri arasında dinamik geçiş sağlanmıştır. Bu yaklaşım, sistemin esnekliğini ve gerçek dünya koşullarına uyumunu artırmıştır.

Sonuç olarak, proje ana hedeflerinin çoğuna başarıyla ulaşılmış; güvenli dosya aktarımı, güvenli iletişim sistemi ve ağ performans analizi gibi çok katmanlı konular başarıyla entegre edilmiştir.

Detaylı anlatım için video: https://www.youtube.com/watch?v=Nlx0FN6iL_c

kaynak kodları için: <https://github.com/ahmetyusufbirdir03/Guvenli-Dosya-Transfer-Sistemi>

KAYNAKLAR

Kaynaklar, alfabetik sırayla ve akademik yazım kurallarına uygun biçimde listelenmiştir.

1. OpenSSL Project. (2024). *OpenSSL: Cryptography and SSL/TLS Toolkit*. <https://www.openssl.org/>
2. Scapy Developers. (2024). *Scapy Documentation*. <https://scapy.readthedocs.io/>
3. Wireshark Foundation. (2024). *Wireshark User Guide*.
https://www.wireshark.org/docs/wsug_html_chunked/
4. The iPerf3 Team. (2024). *iPerf3: A TCP, UDP, and SCTP network bandwidth measurement tool*.
<https://iperf.fr/>
5. Python Software Foundation. (2024). *Python Language Reference, version 3.10*.
<https://docs.python.org/3/>
6. PyCryptodome Project. (2024). *PyCryptodome Documentation*. <https://www.pycryptodome.org/>
7. **GeeksforGeeks. (n.d.)**. *Socket programming in Python*. Retrieved May 30, 2025, from
<https://www.geeksforgeeks.org/socket-programming-python/>