Computer Networks 2 Final Project

PGP (Pretty Good Privacy) Simulation

Hazırlayanlar:

Ersan Ergin

Barışcan İlter

PROBLEM TANIMI

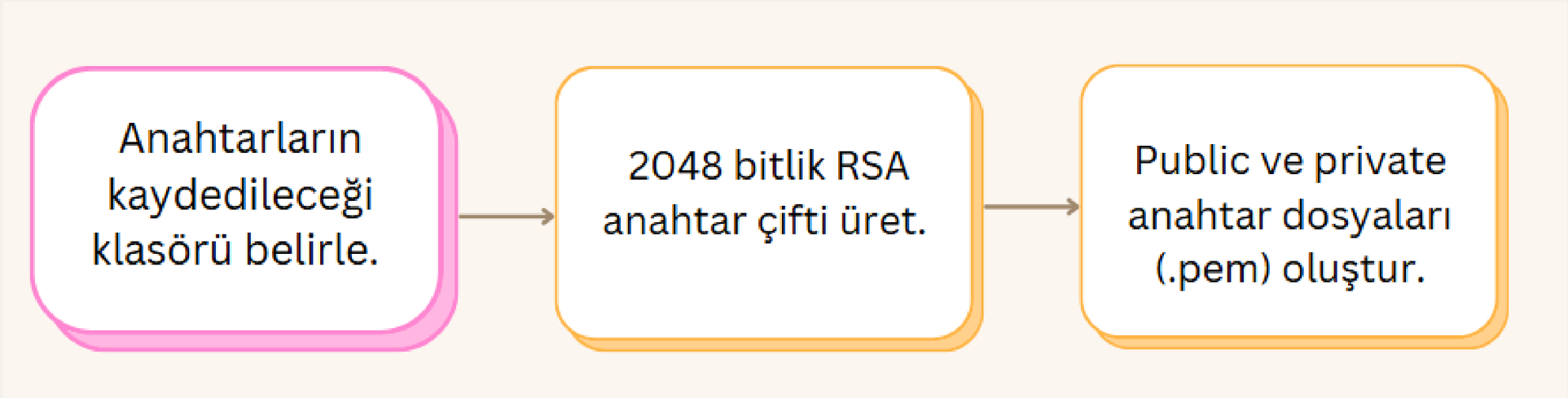
Güvensiz e-posta iletişimi, günümüzün dijital dünyasında önemli bir sorundur. E-posta yoluyla iletilen hassas bilgiler, müdahaleye, yetkisiz erişime veya değişikliğe karşı savunmasızdır. Bu durum, özellikle kişisel, finansal ve kurumsal içeriklerin gizliliği ve bütünlüğü açısından ciddi güvenlik açıklarına yol açar. Geleneksel e-posta sistemleri, mesajları koruyacak güçlü şifreleme ve kimlik doğrulama mekanizmalarına sahip değildir. Bu nedenle, gönderilen bir mesajın yalnızca doğru alıcı tarafından okunması ve içeriğinin bozulmadan ulaşması garanti edilemez.

YAKLAŞIM

- Özetleme ve Dijital İmza: MD5 algoritmasıyla mesajı özetle ve göndericinin private RSA anahtarı ile dijital imza oluştur.
- Compressing: Zlib ile dijital imzalı mesajı sıkıştır.
- Şifreleme: Sıkıştırılmış mesajı rastgele üretilen simetrik anahtar ile IDEA algoritması kullanarak şifrele. Simetrik anahtarı alıcının public RSA anahtarı ile şifrele.
- Encoding: Şifreli anahtar ile şifreli mesajı birleştir. Base64 encoding ile ASCII formatına çevirerek gönderilmeye hazır hale getir.
- Decoding: Base64 decoding ile şifrelenmiş mesajı çöz.
- Çözümleme: Alıcının private RSA anahtarı ile simetrik anahtarı elde et ve mesajı çöz.
- Decompressing: Zlib ile mesaj ve imzayı ayır.
- Doğrulama: MD5 algoritması ile mesajdan elde edilen hash ile dijital imzadan gelen hash arasında karşılaştırma ile doğrulama yap.

SISTEMIN INCELENMESI

Gönderici ve alıcı için anahtar çifti üretimi



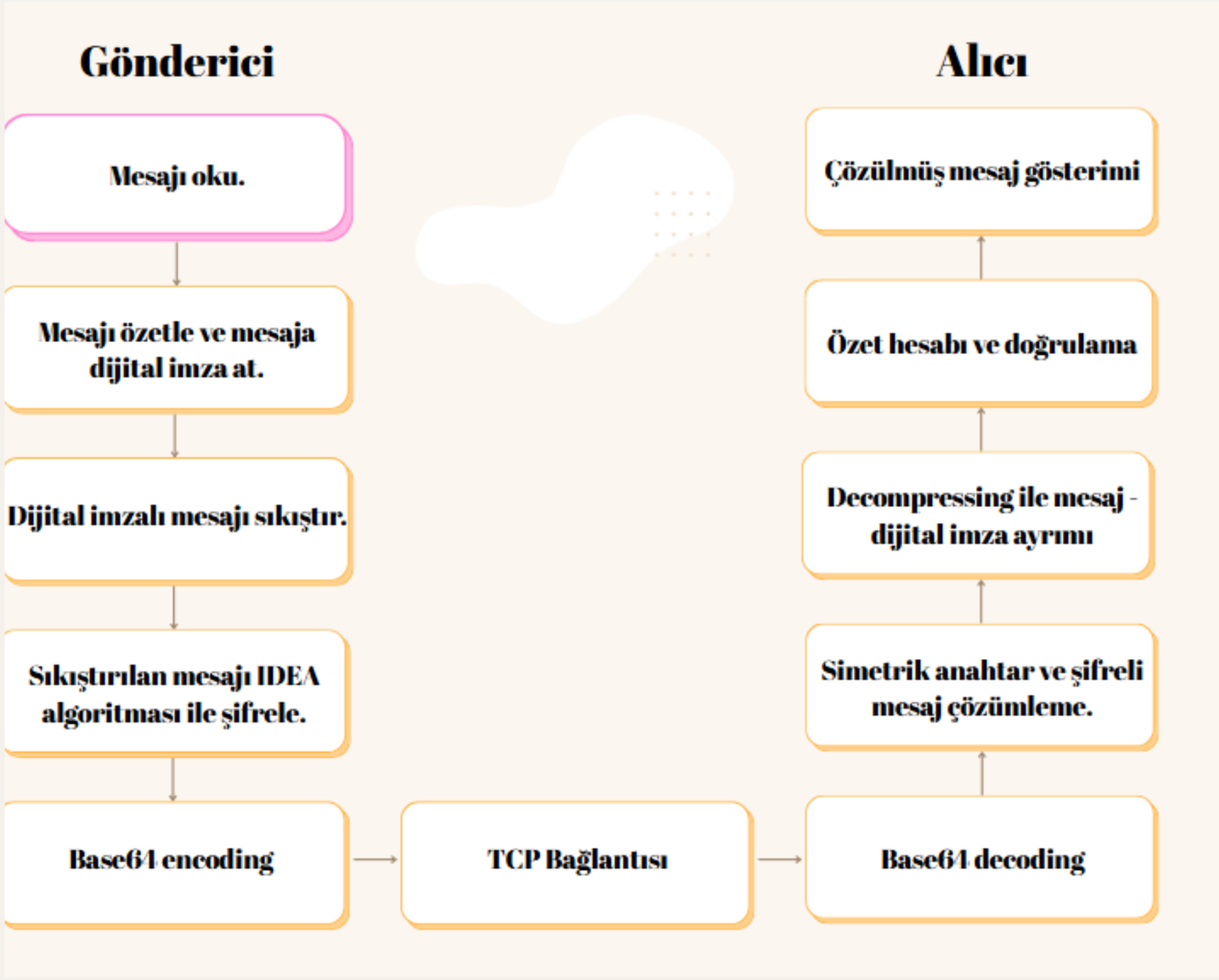
SISTEMIN INCELENMESI (DEVAM)

Gönderici : Mesaj okuma, MD5 özetleme ve dijital imzalama, sıkıştırma, IDEA ile şifreleme, Base64 kodlama

Alıcı : Base64 kod çözümleme, RSA ile simetrik anahtar ve IDEA ile şifreli mesaj çözümleme, mesaj ile dijital imza ayrımı, özet hesabı ve doğrulama, çözülmüş mesaj gösterimi

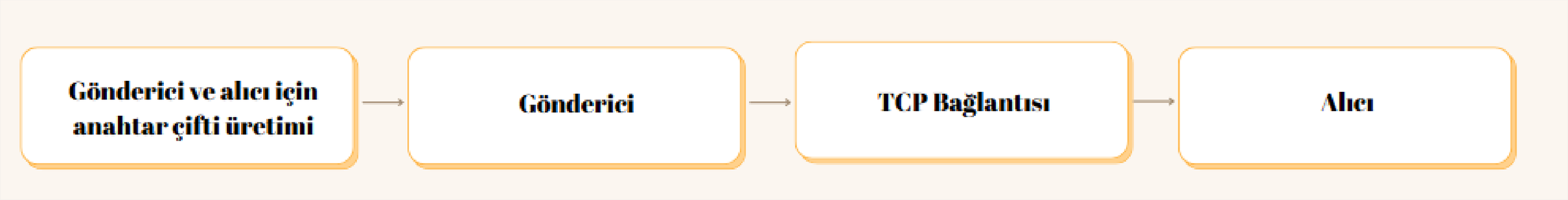
Kullandığımız Metotlar:

- RSA algoritması => Asimetrik şifreleme
- MD5 özetleme ve dijital imzalama
- zlib ile compressing - decompressing
- Base64 encoding ve decoding
- TCP Soket Haberleşmesi



SİMÜLASYONUN GERÇEKLEŞTİRİLMESİ

Bu kısımda PGP simülasyonumuzun nasıl çalıştığını gözlemleyeceğiz.

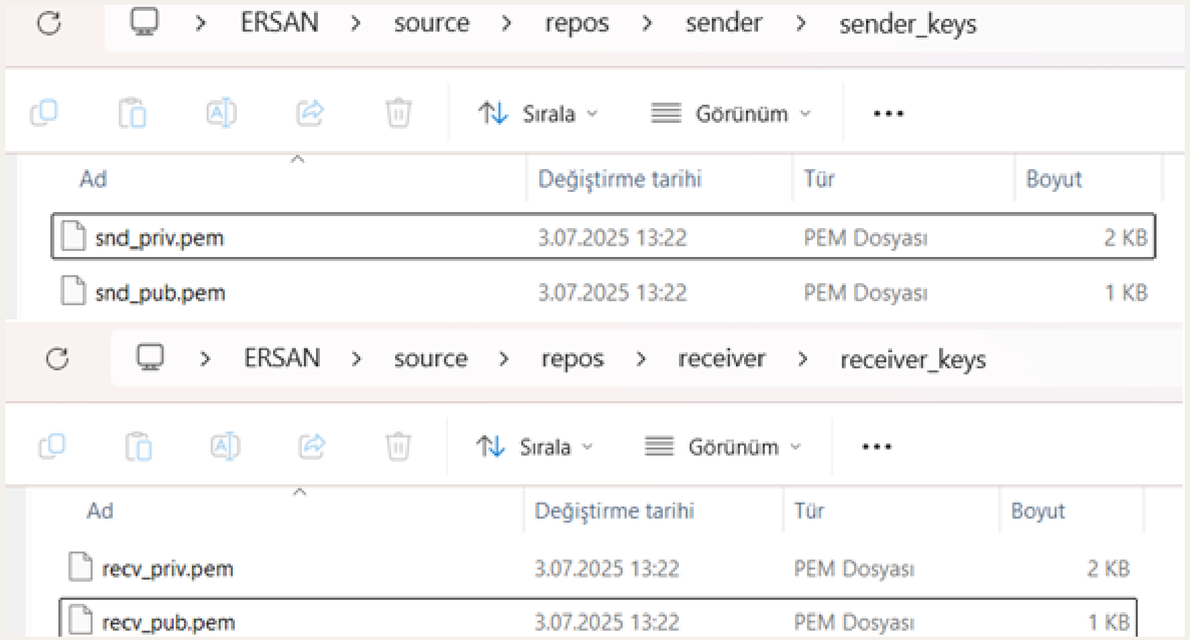


- Mesaj İmzalama ve Şifreleme
- Compressing - Decompressing
- Base64 encoding ve decoding
- Hash hesaplama ve doğrulama

Bu işlemlerin başarılı bir şekilde gerçekleştirildiğini ve simülasyonumuzun başarılı bir şekilde çalıştığını göreceğiz.

```python
import os
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes

# Anahtarların kaydedileceği klasör
key_file = "sender_keys"
os.makedirs(key_file, exist_ok=True)

# RSA 2048-bit key çifti üret
key = RSA.generate(2048)

# PRIVATE KEY: snd_priv.pem
with open(os.path.join(key_file, "snd_priv.pem"), "wb") as f:
    f.write(key.export_key('PEM'))

# PUBLIC KEY: snd_pub.pem
with open(os.path.join(key_file, "snd_pub.pem"), "wb") as f:
    f.write(key.publickey().export_key('PEM'))

print("[✓] Sender için RSA anahtar çifti oluşturulmuştur.")
```

```python
import os
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes

# Anahtarların kaydedileceği klasör
key_file = "receiver_keys"
os.makedirs(key_file, exist_ok=True)

# RSA 2048-bit key çifti üret
key = RSA.generate(2048)

# PRIVATE KEY: recv_priv.pem
with open(os.path.join(key_file, "recv_priv.pem"), "wb") as f:
    f.write(key.export_key('PEM'))

# PUBLIC KEY: recv_pub.pem
with open(os.path.join(key_file, "recv_pub.pem"), "wb") as f:
    f.write(key.publickey().export_key('PEM'))

print("[✓] Receiver için RSA anahtar çifti oluşturulmuştur.")
```



```python
# === PGP Şifreleme ve İmzalama ===
def prepare_pgp_package():
    snd_priv_key = load_private_key(SND_PRIV_KEY)
    recv_pub_key = load_public_key(RECV_PUB_KEY)

    with open(MESSAGE_FILE, "r", encoding="utf-8") as f:
        plaintext_message = f.read()

    # MD5 hash hesapla
    md5_hash = MD5.new()
    md5_hash.update(plaintext_message.encode('utf-8'))
    message_digest = md5_hash.hexdigest()

    # Hash'i gönderenin private key'i ile imzala
    signer = pkcs1_15.new(snd_priv_key)
    signature = signer.sign(MD5.new(plaintext_message.encode('utf-8')))

    # Mesaj + İmza birleştir
    combined_data = plaintext_message + "\n---SIGNATURE---\n" + base64.b64encode(signature).decode()

    # zlib ile sıkıştır
    compressed_data = zlib.compress(combined_data.encode('utf-8'))

    # IDEA için 128-bit (16 byte) rastgele anahtar
    idea_key = get_random_bytes(16)

    # IDEA ile şifrele
    idea_cipher = IDEA(idea_key)
    encrypted_data = idea_cipher.encrypt(compressed_data)

    # IDEA anahtarını alıcının public key'i ile RSA şifrele
    rsa_cipher = PKCS1_OAEP.new(recv_pub_key)
    encrypted_key = rsa_cipher.encrypt(idea_key)

    # Şifrelenmiş anahtar + şifrelenmiş veri birleştir
    final_data = encrypted_key + encrypted_data
```

```python
# Base64 encode
ascii_message = base64.b64encode(final_data).decode()

# Paket bilgilerini kaydet
package = {
    "encrypted_key_size": len(encrypted_key),
    "encrypted_data_size": len(encrypted_data),
    "ascii_message": ascii_message,
    "original_size": len(plaintext_message),
    "compressed_size": len(compressed_data),
    "md5_hash": message_digest
}

with open(PACK_FILE, "w") as f:
    json.dump(package, f, indent=2)

print("[✓] PGP paketi hazırlandı!")
print(f"    Final ASCII mesaj uzunluğu: {len(ascii_message)} karakter")
```

```python
# === TCP sunucu başlatılıyor ===
def start_sender_server():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:
        server.bind((host, port))
        server.listen(1)
        print(f"[ 🍴 ] PGP Sender dinleniyor: {host}:{port}")
        print("Receiver'ın bağlanması bekleniyor...\n")

        conn, addr = server.accept()
        with conn:
            print(f"[ 🟢 ] Receiver bağlandı: {addr}")

            print("[ 📩 ] Sender'ın public key'i gönderiliyor...")
            # Step 1: Sender'ın public key'ini gönder
            with open(SND_PUB_KEY, "rb") as f:
                conn.sendall(f.read() + b"<END_KEY>")

            print("[ 📩 ] Receiver'ın public key'i alınıyor...")
            # Step 2: recv_pub.pem al
            recv_key_data = b""
            while not recv_key_data.endswith(b"<END_KEY>"):
                recv_key_data += conn.recv(1024)
            recv_key_data = recv_key_data.replace(b"<END_KEY>", b"")

            with open(RECV_PUB_KEY, "wb") as f:
                f.write(recv_key_data)
            print("[✓] recv_pub.pem alındı")

            # Step 3: PGP paketini hazırla
            prepare_pgp_package()

            print("[ 📩 ] Şifrelenmiş paket gönderiliyor...")
            # Step 4: encrypted_package.json gönder
            with open(PACK_FILE, "rb") as f:
                conn.sendall(f.read() + b"<END_JSON>")
            print("[✓] encrypted_package.json gönderildi")
```

```
[ 🍴 ] PGP Sender dinleniyor: 127.0.0.1:10000
Receiver'ın bağlanması bekleniyor...

[ 🟢 ] Receiver bağlandı: ('127.0.0.1', 60624)
[ 📩 ] Sender'ın public key'i gönderiliyor...
[ 📩 ] Receiver'ın public key'i alınıyor...
[✓] recv_pub.pem alındı
[✓] PGP paketi hazırlandı!
    Final ASCII mesaj uzunluğu: 1140 karakter
[ 📩 ] Şifrelenmiş paket gönderiliyor...
[✓] encrypted_package.json gönderildi

🎉 PGP mesajı başarıyla gönderildi!
```

```python
# === PGP Çözümleme İşlemi ===
def decrypt_pgp_message():
    with open(PACK_FILE, "r") as f:
        package_data = json.load(f)

    recv_priv_key = load_private_key(PRIV_KEY_FILE)
    snd_pub_key = load_public_key(SND_PUB_KEY_FILE)

    # Base64 decode
    encrypted_combined = base64.b64decode(package_data["ascii_message"])

    # Şifrelenmiş anahtar ve veriyi ayır
    encrypted_key_size = package_data["encrypted_key_size"]
    encrypted_key = encrypted_combined[:encrypted_key_size]
    encrypted_data = encrypted_combined[encrypted_key_size:]

    # RSA ile IDEA anahtarını çöz
    rsa_cipher = PKCS1_OAEP.new(recv_priv_key)
    idea_key = rsa_cipher.decrypt(encrypted_key)

    # IDEA ile veriyi çöz
    idea_cipher = IDEA(idea_key)
    compressed_data = idea_cipher.decrypt(encrypted_data)

    # zlib ile aç
    decompressed_data = zlib.decompress(compressed_data).decode('utf-8')

    # Mesaj ve imzayı ayır
    parts = decompressed_data.split("\n---SIGNATURE---\n")
    if len(parts) != 2:
        raise ValueError("Mesaj formatı hatalı!")

    original_message = parts[0]
    signature_b64 = parts[1]
    signature = base64.b64decode(signature_b64)
```

```python
    # Mesajın MD5 hash'ini hesapla
    md5_hash = MD5.new()
    md5_hash.update(original_message.encode('utf-8'))
    calculated_hash = md5_hash.hexdigest()
    stored_hash = package_data["md5_hash"]
    if calculated_hash == stored_hash:
        print("    ✓ MD5 hash doğrulandı!")
    else:
        print("    ❌ MD5 hash uyuşmuyor!")

    # İmzayı doğrula
    try:
        verifier = pkcs1_15.new(snd_pub_key)
        verifier.verify(MD5.new(original_message.encode('utf-8')), signature
        print("    ✓ Dijital imza geçerli!")
        signature_valid = True
    except (ValueError, TypeError):
        print("    ❌ Dijital imza geçersiz!")
        signature_valid = False

    # Çözülmüş mesajı kaydet
    with open(DECRYPT_OUT, "w", encoding="utf-8") as f:
        f.write(original_message)

    print("\n" + "="*60)
    print("🔓 PGP MESAJI BAŞARIYLA ÇÖZÜLDÜ")
    print("="*60)
    print("\n📨 Mesaj İçeriği:")
    print("-" * 60)
    print(original_message)
    print("-" * 60)

    return signature_valid and (calculated_hash == stored_hash)
```

```python
# === TCP üzerinden sender'a bağlan ===
def connect_to_sender():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((host, port))
        print(f"[ 🔌 ] Sender'a bağlandı: {host}:{port}")

        print("[ 📩 ] Sender'ın public key'i alınıyor...")
        # Step 1: snd_pub.pem al
        sender_key_data = b""
        while not sender_key_data.endswith(b"<END_KEY>"):
            sender_key_data += s.recv(1024)
        sender_key_data = sender_key_data.replace(b"<END_KEY>", b"")

        with open(SND_PUB_KEY_FILE, "wb") as f:
            f.write(sender_key_data)
        print("[✓] snd_pub.pem alındı")

        print("[ 📩 ] Receiver'ın public key'i gönderiliyor...")
        # Step 2: recv_pub.pem gönder
        with open(PUB_KEY_FILE, "rb") as f:
            s.sendall(f.read() + b"<END_KEY>")
        print("[✓] recv_pub.pem gönderildi")

        print("[ 📩 ] Şifrelenmiş paket alınıyor...")
        # Step 3: encrypted_package.json al
        package_data = b""
        while not package_data.endswith(b"<END_JSON>"):
            package_data += s.recv(1024)
        package_data = package_data.replace(b"<END_JSON>", b"")

        with open(PACK_FILE, "wb") as f:
            f.write(package_data)
        print("[✓] encrypted_package.json alındı")

    print("\n" + "="*50)
    print("🔍 PGP ÇÖZÜMLEME BAŞLATILIYOR")
    print("="*50)
```

```python
# Step 4: PGP mesajını çöz
success = decrypt_pgp_message()

if success:
    print("\n🎉 Tüm güvenlik kontrolleri başarılı!")
else:
    print("\n⚠️  Güvenlik kontrolleri başarısız! Mesaj tehlikeye girmiş olabilir.")
```

```
=== PGP RECEIVER ===
[ 🔌 ] Sender'a bağlandı: 127.0.0.1:10000
[ 📩 ] Sender'ın public key'i alınıyor...
[✓] snd_pub.pem alındı
[ 📩 ] Receiver'ın public key'i gönderiliyor...
[✓] recv_pub.pem gönderildi
[ 📩 ] Şifrelenmiş paket alınıyor...
[✓] encrypted_package.json alındı

==================================================
🔍 PGP ÇÖZÜMLEME BAŞLATILIYOR
==================================================
    ✓ MD5 hash doğrulandı!
    ✓ Dijital imza geçerli!

============================================================
🔓 PGP MESAJI BAŞARIYLA ÇÖZÜLDÜ
============================================================

📨 Mesaj İçeriği:
------------------------------------------------------------
Merhaba Barış,


Koskocaman bir 4 yılı geride bıraktık. Bu yıl mezun oluyoruz. İyisiyle ve kötüsüyle dolu dolu bir üniversite
hayatı geçirdiğimizi düşünüyorum. Yepyeni bir macera bizleri bekliyor olacak. Artık meslek hayatına atılma
zamanı geldi. Umarım sevdiğimiz işi bulur ve bu işte daha da ilerilere gideriz. Sana yeni hayatında başarılar
diliyorum.

Sevgilerimle,
Ersan
------------------------------------------------------------

🎉 Tüm güvenlik kontrolleri başarılı!
```

Dinlediğiniz için teşekkür ederiz.