BIL470 - Kriptografi ve Bilgisayar Güvenliği

Dönem Projesi

Rapor

OĞUZHAN AGKUŞ
161044003

Araştırma

Soru: OSI referans modelinin farklı katmanlarında kriptografik protokollerin uygulamasının avantajları ve dezavantajları nelerdir?

Bilgisayar ağlarında çok sayıda güvenlik açığı bulunmaktadır. Bu nedenle, aktarım sırasında veriler saldırılara karşı oldukça savunmasızdır. Bir saldırgan, iletişim kanalını hedef alabilir, verileri elde edebilir ve aynı şeyi okuyabilir veya kötü amaçlarına ulaşmak için yanlış bir mesajı yeniden ekleyebilir. Ağ güvenliği sadece iletişim zincirinin her iki ucundaki bilgisayarların güvenliği ile ilgilenmez, tüm ağın güvenli olmasını sağlamayı amaçlar.

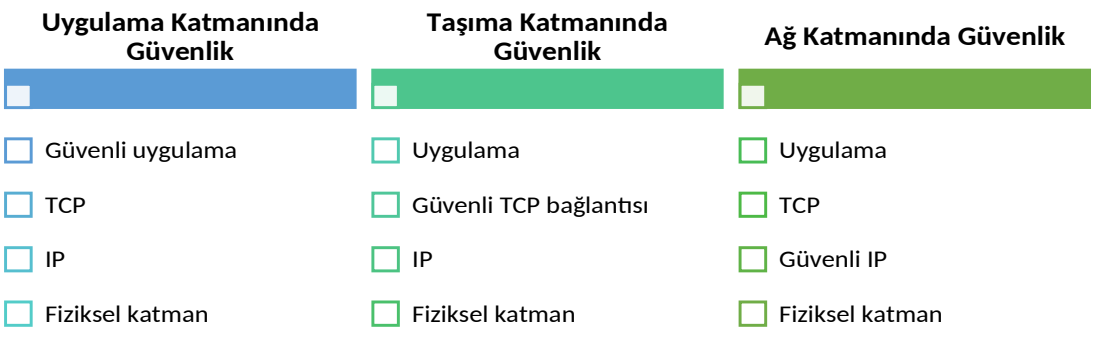
Ağ güvenliği, ağın ve verilerin kullanılabilirliğini, güvenilirliğini, bütünlüğünü ve güvenliğini korumayı gerektirir. Etkili ağ güvenliği, bir ağa girmekten veya ağa yayılmaktan kaynaklanan çeşitli tehditleri ortadan kaldırır. Ağ güvenliğinin birincil amacı gizlilik, bütünlük ve erişilebilirliktir. Ağ güvenliğinin bu üç ayağı genellikle CIA üçgeni olarak temsil edilir. OSI ağ katmanı modelinin belirli bir katmanında çalışabilecek şekilde çeşitli güvenlik mekanizmaları geliştirilmiştir.

Uygulama Katmanı: Bu katmanda kullanılan güvenlik önlemleri uygulamaya özeldir. Farklı uygulama türleri için ayrı güvenlik önlemleri gerekir. Uygulama katmanı güvenliğini sağlamak için uygulamaların değiştirilmesi gerekir. Kriptografik olarak sağlam bir uygulama protokolü tasarlamanın çok zor olduğu ve onu düzgün bir şekilde uygulamanın daha da zor olduğu düşünülmektedir. Bu nedenle, ağ iletişimlerini korumaya yönelik uygulama katmanı güvenlik mekanizmaları, yalnızca bir süredir kullanımda olan standartlara dayalı çözümler olarak tercih edilmektedir.

Uygulama katmanı güvenlik protokolüne bir örnek, e-posta mesajlarını şifrelemek için yaygın olarak kullanılan Güvenli Çok Amaçlı İnternet Posta Uzantılarıdır (S / MIME). DNSSEC, bu katmanda DNS sorgu mesajlarının güvenli alışverişi için kullanılan başka bir protokoldür.

Taşıma Katmanı: Bu katmandaki güvenlik önlemleri, verileri iki ana bilgisayar arasındaki tek bir iletişim oturumunda korumak için kullanılabilir. Taşıma Katmanı Güvenliği protokolünü korumak için en yaygın kullanım HTTP ve FTP oturum trafiğidir. Taşıma Katmanı Güvenliği (TLS) ve Güvenli Soket Katmanı (SSL), bu amaçla kullanılan en yaygın protokollerdir.

Ağ Katmanı: Bu katmandaki güvenlik önlemleri tüm uygulamalara uygulanabilir. Bu nedenle uygulamaya özel değildirler. İki ana bilgisayar veya ağ arasındaki tüm ağ iletişimi, herhangi bir uygulamayı değiştirmeden bu katmanda korunabilir. Bazı ortamlarda, İnternet Protokolü Güvenliği (IPsec) gibi ağ katmanı güvenlik protokolleri, tek tek uygulamalara kontrol eklemedeki zorluklar nedeniyle aktarım veya uygulama kontrol katmanından çok daha iyi bir çözüm sağlar. Bununla birlikte, bu katmandaki güvenlik protokolleri, bazı uygulamaların gerektirebileceği daha az iletişim esnekliği sağlar.



Ayrıca daha yüksek bir katmanda çalışmak üzere tasarlanmış bir güvenlik mekanizması, daha düşük katmanlardaki veriler için koruma sağlayamaz, çünkü daha düşük katmanlar, daha yüksek katmanların farkında olmadığı işlevleri yerine getirir. Bu nedenle, ağ güvenliğini artırmak için birden fazla güvenlik mekanizmasının devreye alınması gerekli olabilir.

Katman	Haberleşme Protokolleri	Güvenlik Protokolleri
Uygulama katmanı (7. Katman)	HTTP, FTP, SMTP	HTTPS, PGP, S/MIME
Taşıma katmanı (4. Katman)	TCP, UDP	SSL, TLS, SSH
Ağ katmanı (3. Katman)	IP	IPsec

Uygulama Katmanında Güvenlik

Günümüzde çeşitli istemci-sunucu uygulamaları çevrimiçi olarak sunulmaktadır. En yaygın olanları ise web uygulamaları ve e-posta hizmetleridir. Her iki uygulamada da müşteri, belirlenen sunucu ile iletişim kurar ve hizmetleri alır. Herhangi bir sunucu üzerinden bir servisi kullanırken, istemci ve sunucu internet üzerinde birçok bilgi alışverişinde bulunur. Bu bilgi işlemleri çeşitli saldırılara açıktır.

Saldırılara karşı ağ güvenliği, verilerin bir ağ üzerinde geçiş halindeyken güvenliğinin sağlanmasını gerektirir. Bu amaca ulaşmak için birçok gerçek zamanlı güvenlik protokolü tasarlanmıştır. Bu tür bir protokol en azından aşağıdaki birincil hedefleri sağlamalıdır:

- Taraflar, birbirlerinin kimliğini doğrulamak için etkileşimli olarak pazarlık yapabilmeliler
- Ağ üzerinde bilgi alışverişi yapmadan önce gizli bir oturum anahtarı oluşturulmalıdır
- Bilgileri şifrelenmiş şekilde transfer edilmelidir

E-postaların güvenliğini sağlama süreci, iletişimin uçtan uca güvenliğini sağlar. Gizlilik, gönderen kimlik doğrulama, mesaj bütünlüğü ve inkâr etmeme güvenlik hizmetleri sağlar. E-posta güvenliği için iki şema geliştirilmiştir: PGP ve S / MIME. Her iki şema da gizli anahtar ve açık anahtar şifreleme kullanır.

Standart DNS araması, sahtekarlık, DNS / önbellek zehirlenmesi gibi saldırılara karşı savunmasızdır. DNS aramasının güvenliğini sağlamak, açık anahtar şifrelemesini kullanan DNSSEC kullanımıyla mümkündür.

Taşıma Katmanında Güvenlik

Saldırılara karşı ağ güvenliği, verilerin bir ağ üzerinde geçiş halindeyken güvenliğinin sağlanmasını gerektirir. Bu amaca ulaşmak için birçok gerçek zamanlı güvenlik protokolü tasarlanmıştır.

Fiziksel ve veri bağlantı katmanları tipik olarak kullanıcı terminalinde ve ağ kartında uygulanır. İşletim sisteminde TCP ve IP katmanları uygulanır. TCP/IP'nin üzerindeki her şey kullanıcı uygulamaları seviyesinde olarak uygulanır.

Faydalar:

- Uygulamalara karşı şeffaftır.
- Sunucunun kimliği doğrulanır.
- Katmana ait başlıklar gizlidir.

Sınırlamalar:

- Yalnızca TCP tabanlı uygulamalar için geçerlidir (UDP değil).
- TCP/IP başlıkları ortadadır.
- İstemci ve sunucu arasında doğrudan iletişim için uygundur.

- İstemci kimlik doğrulaması isteğe bağlı olduğundan SSL, inkâr edilemezlik sağlamaz.
- Gerekirse, SSL istemci kimlik doğrulamasının üzerinde gerçeklemek gerekir.

Bir kullanıcının oturumu sırasında olası saldırıları azaltmanın bir yolu, güvenli bir iletişim protokolü kullanmaktır. Bu tür iletişim protokollerinden ikisi, Güvenli Soket Katmanı (SSL) ve Aktarım Katmanı Güvenliği (TLS) bu bölümde tartışılmaktadır. Bu protokollerin her ikisi de taşıma katmanında çalışır.

Telnet'in yerini almak üzere tasarlanan başka bir aktarım katmanı protokolü olan Secure Shell (SSH), uzaktan oturum açma olanağı için güvenli araçlar sağlar. Secure Command Shell ve SFTP gibi çeşitli hizmetleri sunabilir.

Taşıma katmanında güvenliğin sağlanmasının birçok faydası vardır. Ancak bu katmanda tasarlanan güvenlik protokolü yalnızca TCP ile kullanılabilir. UDP kullanılarak gerçekleştirilen iletişim için güvenlik sağlamazlar.

Ağ Katmanında Güvenlik

Ağ katmanı güvenlik kontrolleri, özellikle internet gibi paylaşılan ağlar üzerinden iletişimin güvenliğini sağlamak için sıklıkla kullanılmaktadır. Bunun sebebi birçok uygulama için tek aşamada onları değiştirmeden koruyabilirler.

Bu protokollerin çoğu, standart internet protokolünün (IP) doğasında bulunan güvenlik eksikliğini telafi etmek için OSI protokol yığınının daha yüksek katmanlarına odaklanmış durumda kaldı. Oldukça başarılı olsa da bu yöntemler herhangi bir uygulamada kullanılmak üzere kolayca genelleştirilemez. Örneğin SSL, HTTP veya FTP gibi uygulamaları güvenli hale getirmek için özel olarak geliştirilmiştir. Ancak güvenli iletişime ihtiyaç duyan birkaç başka uygulama da vardır.

Bu ihtiyaç, IP katmanında bir güvenlik çözümü geliştirmeye yol açtı, böylece tüm üst katman protokolleri bundan yararlanabildi. 1992'de IETF standart bir “IPsec” tanımlamaya başladı.

IPsec, ağ bağlantılarının güvenliğini sağlamak için bir protokoller paketidir. Oldukça karmaşık bir mekanizmadır, çünkü belirli bir şifreleme algoritması ve kimlik doğrulama işlevinin açık bir tanımını vermek yerine, her iki iletişim ucunun üzerinde anlaştığı herhangi bir şeyin uygulanmasına izin veren bir çerçeve sağlar.

Taşıma modu, IP başlığını değiştirmeden iki uç nokta arasında güvenli bir bağlantı sağlar. Tünel modu, IP paket içeriğinin tamamını kapsar. Yeni IP başlığı ekler. İkincisi, güvenilmeyen bir ağ üzerinden sanal bir güvenli tünel sağladığı için geleneksel bir VPN oluşturmak için kullanılır.

Bir IPsec bağlantısı kurmak neredeyse bütün şifreleme yöntemlerini kapsayabilir. Kimlik doğrulama genellikle MD5 veya SHA-1 gibi kriptografik özüt alma işlemleri üzerine kurulur. Şifreleme algoritmaları DES, 3DES, Blowfish ve AES'dir. Diğer algoritmalar da mümkündür.

İletişim kuran her iki uç noktanın şifrelemede kullanılan gizli değerleri bilmesi gerekir. Manuel anahtarlar, muhtemelen bazı bant dışı mekanizmalarla iletilen gizli değerlerin her iki uçta manuel olarak girilmesini gerektirir ve IKE (İnternet Anahtar Değişimi), bunu çevrimiçi yapmak için gelişmiş bir mekanizmadır.

Programlama

Kodlama kısmında AES şifreleme yöntemini herhangi bir kütüphane ya da API kullanmadan, sadece Python dili kullanarak temel düzeyde gerçekledim. Daha sonra bu temel gerçeklemeyi daha sonra CBC (şifre-bloku zincirleme) ve OFB (çıktı geri bildirim) modlarında çalışabilecek şekilde genişlettim.

Ek olarak CBC modunu özüt alma metodu olarak kullanarak, dosya bütünlüğünü sağlamayı amaçlayan ve bunu kontrol eden bir araç geliştirdim.

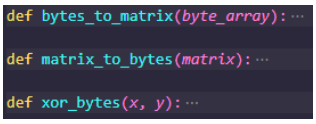
Aşağıda bu gerçeklemelerin önemli detaylarına yer verip, çeşitli test sonuçlarına ait ekran görüntülerini paylaştım. Ayrıca kod içinde gerekli açıklamalara yorum satırı olarak yer verdim.

Kaynak kodu klasöründe yer alan dosyalar şöyledir:

- aes.py → AES algoritmasını temsil eden ve gerçekleyen bir sınıf içerir. Kütüphane olarak eklenip kullanmalıdır. Üç farklı anahtar uzunluğunu destekler. Üç farklı modda çalıştırılması mümkündür: ECB, CBC, OFB
- test.py → Gerçeklenen AES algoritması farklı modlarda, farklı anahtar uzunluklarıyla, farklı düz metinler üzerinde test edilir. Kod çalıştırıldığında testler otomatik olarak çalışır.
- checker.py → Dosyaların bütünlüğünün sağlanmasını amaçlayan araç. Komut satırı uygulaması olarak çalışır. -h parametresi ile çalıştırılınca nasıl çalıştırılması gerektiği hakkında bilgi verir. 3 farklı işlem yapabilir. Verilen dosyanın özütünü alarak dosyanın sonuna ekler. Başka bir kullanıcı bu dosyayı bu araçla eklenen özütün geçerli olup olmadığını test eder. Eğer özüt değerleri uyuşmazsa dosyanın değiştiği anlamı çıkarılır. Ek olarak eklenen özütün silinmesi de mümkündür.
- key_file → checker.py aracının kullanacağı anahtarı içerir. 16, 24 veya 32 bayt uzunluğunda olmalıdır. Aksi taktirde hata verecektir. Dosyayı imzalayan ve kontrolünü yapan kişilerde bu anahtar bilgisi olmalıdır.

AES Gerçeklemesi

Algoritma 4x4 boyutunda matrisler tabanında yaptığı için bu dönüşümleri yapan ve bu bloklara arasında xor işlemi yapan fonksiyonlara ihtiyaç duydum.



S-box gibi yapıları tanımlarken AES'in Wikipedia sayfasını baz aldım. Bunlar küresel değişken olarak saklanmakta.

AES sınıfı içinde algoritmanın adımlarını gerçekleştiren fonksiyonlar tanımladım. Şifre çözme işlemi sırasında bu fonksiyonların terslerine ihtiyaç duyulduğu için, bu amacı yerine getiren ters fonksiyonları ekledim.

```
def _add_round_key(self, block, key): ···

def _sub_bytes(self, block): ···

def _inv_sub_bytes(self, block): ···

def _shift_rows(self, block): ···

def _inv_shift_rows(self, block): ···

def _mix_single_column(self, column): ···

def _mix_columns(self, block): ···

def _inv_mix_columns(self, block): ···
```

CBC gibi modlarda şifrelenecek verinin uzunluğunun 16'nın yani tur anahtarı uzunluğunun tam katı olması gerekmektedir. Eğer tam katı değilse bunu uygun hale getirmek için belli standartlara göre ekleme yapılır. Benim kodladığım ekleme fonksiyonunda PKCS7 standardını varsaydım. Şifreleme işlemi başlamadan yapılan bu ekleme işlemine karşılık olarak, şifre çözme işlemi sonunda eklenen bu verilerin kaldırılması gerekir.

```
def _add_padding(self, plain_text): ···

def _remove_padding(self, plain_text): ···
```

AES nesnesi oluşturulurken bir anahtara ihtiyaç duyar. Bu anahtarı temel alarak tur anahtarları üretilir. Geçersiz uzunlukta bir anahtar verilirse, hata verilip çıkış yapması sağlanır.

```
def __init__(self, key):
    """
    Initializes with given key.
    Generates round keys.
    """
    self._key_size = len(key)
    if self._key_size in self._rounds_by_key_size:
        self._key = key
        self._rounds = self._rounds_by_key_size[self._key_size]
        self._round_keys = self._generate_round_keys(key)
    else:
        print("Invalid key length!")
        sys.exit(1)
```

AES blok tabanlı bir şifreleme yöntemi olduğu için bir bloğun şifreleme işlemi tanımlanırsa daha sonra bu işlem sürekli tekrar edilebilir. İlk önce 16 bayt uzunluğundaki düz metin 4x4 matrise dönüştürülür. Daha son ilk tur anahtarı eklenir. Ardından s-box'tan geçirme, satırları kaydırma, sütunları karıştırma ve tur anahtarı ekleme işlemi. Tur anahtarı sayısının 1 eksiği kadar devam eder. Son adımda ise bu döngüdeki sütunları karıştırma işlemi uygulanmaz. Şifre çözme işleminde ise bu işlemler ters sırada yapılır.

```
def encrypt_block(self, plain_text):
    plain_state = bytes_to_matrix(plain_text)
    self._add_round_key(plain_state, self._round_keys[0])

    for i in range(1, self._rounds):
        self._sub_bytes(plain_state)
        self._shift_rows(plain_state)
        self._mix_columns(plain_state)
        self._add_round_key(plain_state, self._round_keys[i])

    self._sub_bytes(plain_state)
    self._shift_rows(plain_state)
    self._add_round_key(plain_state, self._round_keys[-1])

    return matrix_to_bytes(plain_state)
```

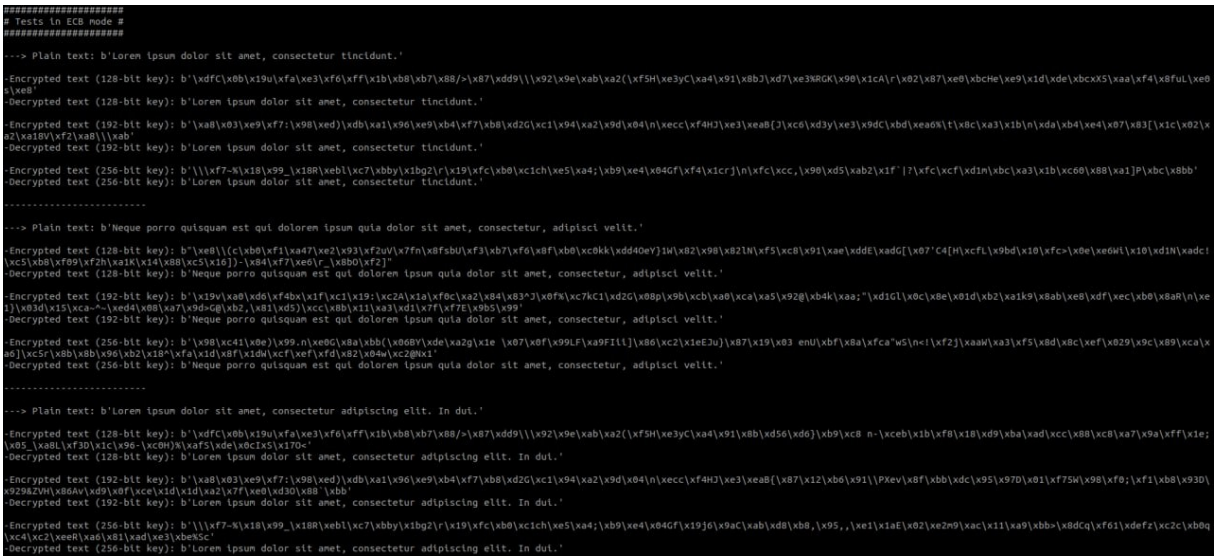
```
def decrypt_block(self, cipher_text):
    cipher_state = bytes_to_matrix(cipher_text)

    self._add_round_key(cipher_state, self._round_keys[-1])
    self._inv_shift_rows(cipher_state)
    self._inv_sub_bytes(cipher_state)

    for i in range(self._rounds - 1, 0, -1):
        self._add_round_key(cipher_state, self._round_keys[i])
        self._inv_mix_columns(cipher_state)
        self._inv_shift_rows(cipher_state)
        self._inv_sub_bytes(cipher_state)

    self._add_round_key(cipher_state, self._round_keys[0])

    return matrix_to_bytes(cipher_state)
```

Bu blok şifreleme işlemi farklı birtakım işlemler eklenerek farklı modlara dönüştürülebilir. Bu noktada ECB, CBC ve OFB modlarının gerçeklemesini yaptım.

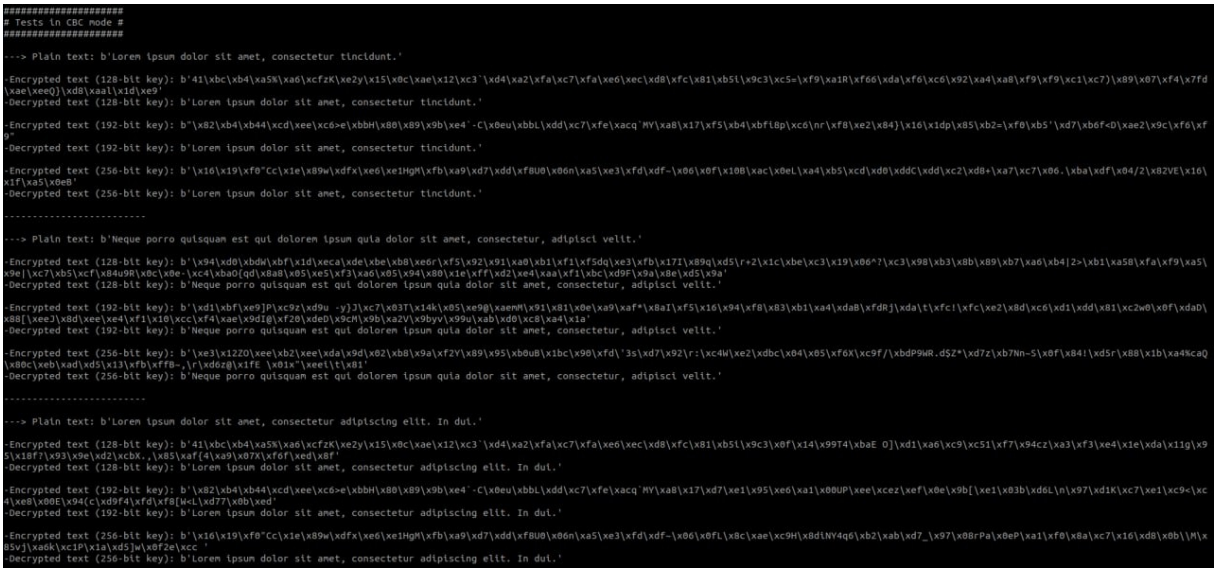
Test dosyasında ise 3 farklı rastgele metin, 3 farklı uzunlukta rastgele anahtar ve 1 rastgele başlangıç vektörü oluşturulmuştur. Her metin 3 farklı anahtarla şifrelenip daha sonra deşifre edilmiştir.

```
plain_text_1 = "Lorem ipsum dolor sit amet, consectetur tincidunt.".encode()
plain_text_2 = "Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.".encode()
plain_text_3 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.".encode()

iv = os.urandom(16)
key_128 = os.urandom(16)
key_192 = os.urandom(24)
key_256 = os.urandom(32)

object_1 = AES(key_128)
object_2 = AES(key_192)
object_3 = AES(key_256)
```

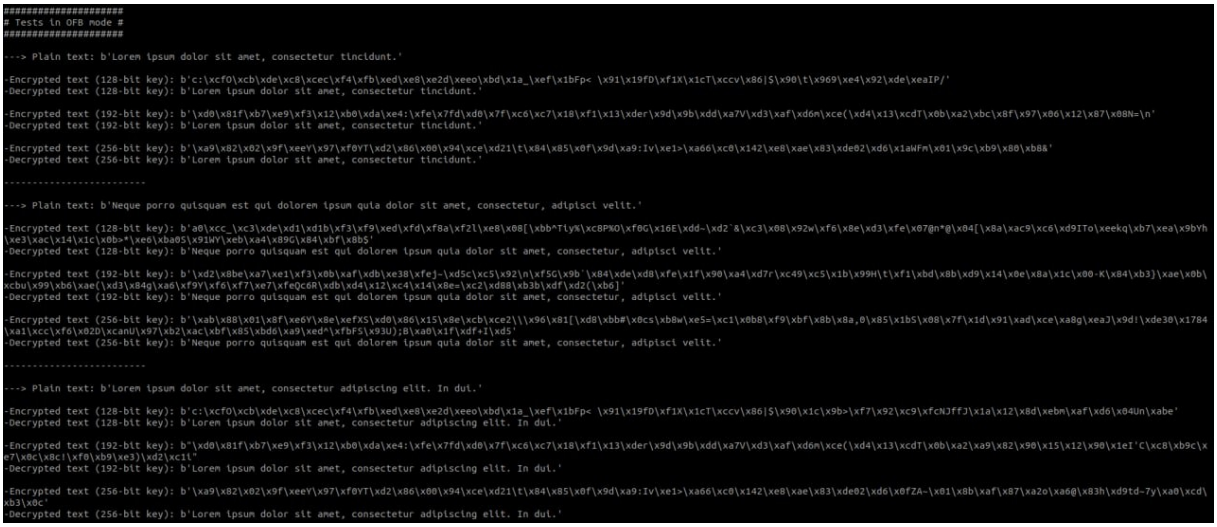
Her mod için uygulanan testlerin ekran çıktıları bu şekildedir:



```
####################
# Tests in ECB mode #
####################

---> Plain text: b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

-Encrypted text (128-bit key): b'\xdfC\x0b\x19u\xfa\xe3\xf6\xff\x1b\xb8\xb7\x88/>\x87\xdd9\\\x92\x9e\xab\xa2(\xf5H\xe3yC\xa4\x91\x8bJ\xd7\xe3%RGK\x90\x1cA\r\x02\x87\xe0\xbcHe\xe9\x1d\xde\xbcxX5\xaa\xf4\x8fuL\xe8
s\xe8'
-Decrypted text (128-bit key): b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

-Encrypted text (192-bit key): b'\xa8\x03\xe9\xf7:\x98\xed)\xdb\xa1\x96\xe9\xb4\xf7\xb8\xdzG\xc1\x94\xa2\x9d\x04\n\xecc\xf4HJ\xe3\xeaB{J\xc6\xd3y\xe3\x9dC\xbd\xea6%\t\x8c\xa3\x1b\n\xda\xb4\xe4\x07\x83['\x1c\x02\x
a2\xa18V\xf2\xa8\\\xab'
-Decrypted text (192-bit key): b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

-Encrypted text (256-bit key): b'\\\xf7~%\x18\x99_\x18R\xebl\xc7\xbby\x1bg2\r\x19\xfc\xb0\xc1ch\xe5\xa4;\xb9\xe4\x04Gf\xf4\x1crj\n\xfc\xcc,\x90\xd5\xab2\x1f`|?\xfc\xcf\xd1m\xbc\xa3\x1b\xc60\x88\xa1]P\xbc\x8bb'
-Decrypted text (256-bit key): b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

------------------------

---> Plain text: b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

-Encrypted text (128-bit key): b"\xe8\|(c\xb0\xf1\xa47\xe2\x93\xf2uV\x7fn\x8fsbU\xf3\xb7\xf6\x8f\xb0\xc0kk\xdd4OeY}1W\x82\x98\x82lN\xf5\xc8\x91\xae\xddE\xadG[\x07'C4[H\xcfL\x9bd\x10\xfc>\x0e\xe6Wi\x10\xd1N\xadc!
\xc5\xb8\xf09\xf2h\xa1K\x14\x88\xc5\x16]}-\x84\xf7\xe6\r_\x8b0\xf2]"
-Decrypted text (128-bit key): b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

-Encrypted text (192-bit key): b'\x19v\xa0\xd6\xf4bx\x1f\xc1\x19:\xc2A\x1a\xf0c\xa2\x84\x83^J\x0f%\xc7kC1\xd2G\x08p\x9b\xcb\xa0\xca\xa5\x92@\xb4k\xaa;"\xd1Gl\x0c\x8e\x01d\xb2\xa1k9\x8ab\xe8\xdf\xec\xb0\x8aR\n\xe
1}\x03d\x15\xca-^~\xed4\x08\xa7\x9d>G@\xb2,\x81\xd5)\xcc\x8b\x11\xa3\xd1\x7f\xf7E\x9b5\x99'
-Decrypted text (192-bit key): b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

-Encrypted text (256-bit key): b'\x98\xc41\x0e)\x99.n\xe0G\x8a\xbb{\x06BY\xde\xa2g\x1e \x07\x0f\x99LF\xa9FIli]\x86\xc2\x1eEJu}\x87\x19\x03 enU\xbf\x8a\xfca"wS\n<!\xf2j\xaaW\xa3\xf5\x8d\x8c\xef\x029\x9c\x89\xca\x
e6]\xc5r\x8b\x8b\x96\xb2\x18^\xfa\x1d\x8f\x1d%\xcf\xef\xfd\x82\x04w\xc2@%x3'
-Decrypted text (256-bit key): b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

------------------------

---> Plain text: b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'

-Encrypted text (128-bit key): b'\xdfC\x0b\x19u\xfa\xe3\xf6\xff\x1b\xb8\xb7\x88/>\x87\xdd9\\\x92\x9e\xab\xa2(\xf5H\xe3yC\xa4\x91\x8b\xd56\xd6}\xb9\xc8 n-\xceb\x1b\xf8\x18\xd9\xba\xad\xcc\x88\xc8\xa7\x9a\xff\x1e;
\x05_\xa8L\xf3D\x1c\x96-\xc0H)%\xafS\xde\x0cIxS\x17O<'
-Decrypted text (128-bit key): b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'

-Encrypted text (192-bit key): b'\xa8\x03\xe9\xf7:\x98\xed)\xdb\xa1\x96\xe9\xb4\xf7\xb8\xdzG\xc1\x94\xa2\x9d\x04\n\xecc\xf4HJ\xe3\xeaB{\x87\x12\xb6\x91\|PXev\x8f\xbb\xdc\x95\x97D\x01\xf75W\x98\xf0;\xf1\xb8\x93D\
x929%ZVH\xB6Av\xd9\x0f\xce\x1d\x1d\xa2\x7f\xe0\xd30\x88``\xbb'
-Decrypted text (192-bit key): b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'

-Encrypted text (256-bit key): b'\\\xf7~%\x18\x99_\x18R\xebl\xc7\xbby\x1bg2\r\x19\xfc\xb0\xc1ch\xe5\xa4;\xb9\xe4\x04Gf\x19j6\x9aC\xab\xd8\xb8,\x95,,\xe1\x1aE\x02\xe2m9\xac\x11\xa9\xbb>\x8dCq\xf61\xdefz\xc2c\xb0q
\xc4\xc2\xeeR\xa6\x81\xad\xe3\xbe%Sc'
-Decrypted text (256-bit key): b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'
```

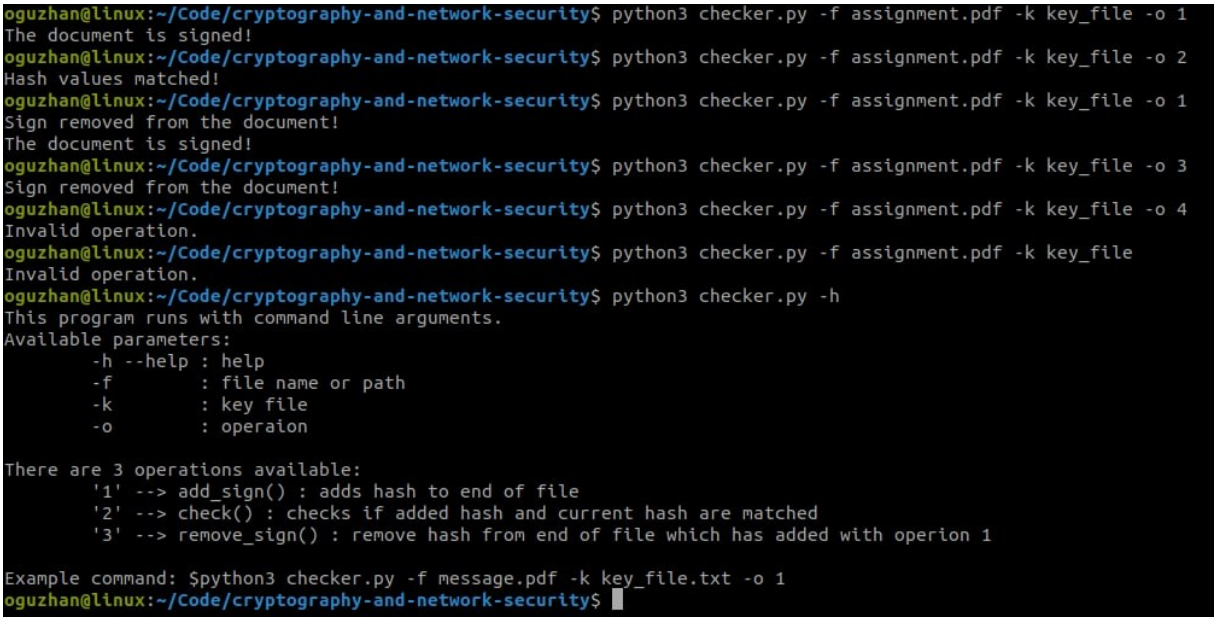


```
####################
# Tests in CBC mode #
####################

---> Plain text: b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

-Encrypted text (128-bit key): b'41\xbc\xb4\xa5%\xa6\xcfzK\xe2y\x15\x0c\xae\x12\xc3`\xd4\xa2\xfa\xc7\xfa\xe6\xec\xd8\xfc\x81\xb5i\x9c3\xc5=\xf9\xa1R\xf66\xda\xf6\xc6\x92\xa4\xa8\xf9\xf9\xc1\xc7)\x89\x07\xf4\x7fd
\xae\xeeQ}\xd8\xaal\x1d\xe9'
-Decrypted text (128-bit key): b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

-Encrypted text (192-bit key): b"\x82\xb4\xb44\xcd\xee\xc6>e\xbbH\x80\x89\x9b\xe4`-C\x0eu\xbbL\xdd\xc7\xfe\xacq`MY\xa8\x17\xf5\xb4\xbfi8p\xc6\nr\xf8\xe2\x84}\x16\x1dp\x85\xb2=\xf0\xb5'\xd7\xb6f<D\xae2\x9c\xf6\xf
9"
-Decrypted text (192-bit key): b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

-Encrypted text (256-bit key): b'\x16\x19\xf0"Cc\x1e\x89w\xdfx\xe6\xe1HgM\xfb\xa9\xd7\xdd\xf8U0\x06n\xa5\xe3\xfd\xdf~\x06\x0f\x10B\xac\x0eL\xa4\xb5\xcd\xd0\xddC\xdd\xc2\xd8+\xa7\xc7\x06.\xba\xdf\x04/2\x82VE\x16\
x1f\xa5\x0eB'
-Decrypted text (256-bit key): b'Lorem ipsum dolor sit amet, consectetur tincidunt.'

------------------------

---> Plain text: b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

-Encrypted text (128-bit key): b'\x94\xd0\xbdW\xbf\x1d\xeca\xde\xbe\xb8\xe0r\xf5\x92\x91\xa0\xb1\xf1\xf5dq\xe3\xfb\x17I\x89q\xd5\r+2\x1c\xbe\xc3\x19\x06^?\xc3\x98\xb3\x8b\x89\xb7\xa6\xb4|2>\xb1\xa58\xfa\xf9\xa5\
x9e|\xc7\xb5\xcf\x84u9R\x0c\x0e-\xc4\xbaO{qd\x8a8\x05\xe5\xf3\xa6\x05\x94\x80\x1e\xff\xd2\xe4\xaa\xf1\xbc\xd9F\x9a\x8e\xd5\x9a'
-Decrypted text (128-bit key): b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

-Encrypted text (192-bit key): b'\xd1\xbf\xe9[P\xc9z\xd9u -yJ]\xc7\x03T\x14k\x05\xe9@\xaemM\x91\x81\x0e\xa9\xaf*\x8aI\xf5\x16\x94\xf8\x83\xb1\xa4\xdaB\xfdRj\xda\t\xfc!\xfc\xe2\x8d\xc6\xd1\xdd\x81\xc2w0\x0f\xdaD\
x88[\xee]\x8d\xee\xe4\xf1\x10\xcc\xf4\xae\x9dI@\xf20\xdeD\x9cM\x9b\xa2V\x9byv\x99u\xab\xd0\xc8\xa4\x1a'
-Decrypted text (192-bit key): b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

-Encrypted text (256-bit key): b'\xe3\x12ZO\xee\xb2\xee\xda\x9d\x02\xb8\x9a\xf2Y\x89\x95\xb0uB\x1bc\x90\xfd'3s\xd7\x92\r:\xc4W\xe2\xdbc\x04\x05\xf6X\xc9f/\xbdP9WR.dSZ*\xd7z\xb7Nn~5\x0f\x84!\xd5r\x88\x1b\xa4%caQ
\x80c\xeb\xad\xd5\x1J\xfb\xff8~,\r\xdoz@\x1fE \x01x"\xeei\t\x81'
-Decrypted text (256-bit key): b'Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit.'

------------------------

---> Plain text: b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'

-Encrypted text (128-bit key): b'41\xbc\xb4\xa5%\xa6\xcfzK\xe2y\x15\x0c\xae\x12\xc3`\xd4\xa2\xfa\xc7\xfa\xe6\xec\xd8\xfc\x81\xb5i\x9c3\x0f\x14\x99T4\xbaE O]\xd1\xa6\xc9\xc51\xf7\x94cz\xa3\xf3\xe4\x1e\xda\x11g\x9
5\x18f7|x93\x9e\xd2\xcbx.,\x85\xaf{4\xa9\x07X\xf6f\xed\x8f'
-Decrypted text (128-bit key): b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'

-Encrypted text (192-bit key): b'\x82\xb4\xb44\xcd\xee\xc6>e\xbbH\x80\x89\x9b\xe4`-C\x0eu\xbbL\xdd\xc7\xfe\xacq`MY\xa8\x17\xd7\xe1\x95\xe6\xa1\x0BUP\xee\xcez\xef\x0e\x9b[\xe1\x03b\xd6L\n\x97\xd1K\xc7\xe1\xc9<\xc
4\xe8\x08E\x94{c\xd9f4\xfd\xf8[W<L\xd77\x0b\xed'
-Decrypted text (192-bit key): b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'

-Encrypted text (256-bit key): b'\x16\x19\xf0"Cc\x1e\x89w\xdfx\xe6\xe1HgM\xfb\xa9\xd7\xdd\xf8U0\x06n\xa5\xe3\xfd\xdf~\x06\x0fL\x8c\xae\xc9H\x8diNY4q6\xb2\xab\xd7_\x97\x08rPa\x0eP\xa1\xf0\x8a\xc7\x16\xd8\x8b\|M\x
85vj\xa6k\xc1P\x1a\xd5]w\x8f2e\xcc '
-Decrypted text (256-bit key): b'Lorem ipsum dolor sit amet, consectetur adipiscing elit. In dui.'
```



Doküman Kontrol Aracı

Bu araç bir AES şifrelemeyi CBC modunda kullanarak doküman verisini şifreleyip, oluşan şifreli metnin son 16 baytını özüt değeri olarak kabul eder. Daha sonra bu özüt değeri dokümanın sonuna ekler. Daha sonra eklenen bu özüt yine bu araçla kontrol edilip dosyanın değiştirilip değiştirilmediği anlaşılabilir. Bu araç komut satırı uygulaması olarak çalışmaktadır ve örnek çalıştırma yöntemi aşağıdaki görüntüde gösterilmiştir.



Dosya imzalandıktan sonra üzerinde değişiklik yapılıp tekrar kontrol edilmiştir. Bu test örneğine ait ekran görüntüsü şöyledir:

```
oguzhan@linux:~/Code/cryptography-and-network-security$ python3 checker.py -f assignment.pdf -k key_file -o 1
The document is signed!
oguzhan@linux:~/Code/cryptography-and-network-security$ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> file_size = os.stat("assignment.pdf").st_size
>>> file = open("assignment.pdf", "rb+")
>>> file.seek(file_size - 19)
352161
>>> file.write("*".encode())
1
>>> file.close()
>>> exit()
oguzhan@linux:~/Code/cryptography-and-network-security$ python3 checker.py -f assignment.pdf -k key_file -o 2
Hash values did not matched!
oguzhan@linux:~/Code/cryptography-and-network-security$
```