Gebze Teknik Üniversitesi
Bilgisayar Mühendisliği

CSE 470 – KRİPTOGRAFİ VE BİLGİSAYAR GÜVENLİĞİ

ÖDEV RAPORU

FURKAN ÖZEV
161044036

1- ARAŞTIRMA KISMI:

OSI temel referans modelinin uygulama (katman 7), ağ (katman 3) ve taşıma (katman 4) katmanlarında kullanılan haberleşme ve kriptografik protokolleri şu şekilde listelenir:

KATMAN	HABERLEŞME PROTOKOLLER	KRİPTOGRAFİK PROTOKOLLER
Uygulama Katmanı	HTTP – FTP – SMTP	HTTPS – S/MIME – PGP
Taşıma Katmanı	TCP – UDP	SSL – TLS – SSH
Ağ Katmanı	IP	IPSec

A- Uygulama Katmanı:

Uygulama katmanı, kullanıcıların gerçekte bilgisayarla iletişime geçtiği yerdir.

Programların ağı kullanabilmesi için araçlar sunar.

Diğer katmanlarda olduğu gibi bir üst katmanı olmadığı için o katmana servis sağlaması gibi bir durum söz konusu değildir.

Uygulama katmanında uygulamaların ağ üzerinde çalışması sağlanır.

OSI modelinde en üst katmandan yola çıkan ham veri, her katmanda o katmanla ilgili bazı ek bilgiler eklenerek bir alt katmana aktarılır.

Bu katmandaki haberleşme protokoller: HTTP – FTP – SNMP - DHCP

Uygulama katmanındaki şifrelemede, uçtan uca güvenlik, istemci çalışma ortamında ve sunucu ana bilgisayarlarında şifreleme uygulamaları ile kullanıcı düzeyinde sağlanır.

Gerekirse, şifreleme kaynağa(encrpytion) ve şifre çözme(decryption) ise hedefe mümkün olduğu kadar yakın olacaktır.

Uygulama katmanı şifrelemesinde yalnızca veriler şifrelenir.

Bu katmandaki kriptografik protokoller: HTTPS – S/MIME – PGP – MSP

Uygulama katmanı son kullanıcıya en yakın katman olduğu için bilgisayar korsanlarına en büyük tehdit yüzeyini sağlar.

Yetersiz uygulama katmanı güvenliği, performans ve kararlılık sorunlarına, veri hırsızlığına ve bazı durumlarda ağın kapatılmasına neden olabilir.

Uygulama katmanı saldırılarına örnek olarak dağıtılmış hizmet reddi saldırıları (DDoS) saldırıları, HTTP flood'leri, SQL enjeksiyonları, siteler arası komut dosyası oluşturma, parametre değiştirme ve Slowloris saldırıları verilebilir .

Kriptografik protokolleri: **HTTPS – S/MIME – PGP**

1- HTTPS:

HTTPS (HTTP Secure, Güvenli Hiper Metin Aktarım İletişim Protokolü) Http'nin çok gelişmiş ve güvenli bir sürümüdür.

HTTPS'te, iletişim protokolü Taşıma Katmanı Güvenliği (TLS) veya öncesinde, onun öncülü olan Güvenli Soket Katmanı (SSL) ile şifrelenir.

Bu nedenle protokol sık sık TLS üzerinden HTTP veya SSL üzerinden HTTP olarak da adlandırılır.

Tüm iletişimi SSL ile şifreleyerek birçok güvenli işlemin oluşturulmasına izin verir. HTTPS, özünde SSL/TLS ve http'nin bir kombinasyonudur.

Ağ sunucusunun şifreli ve güvenli tanımlanmasını sağlar.

Ayrıca sunucu ve tarayıcı arasında güvenli bir şifreli bağlantıya izin verir.

Çift yönlü veri güvenliği sunar. Bu, potansiyel olarak gizli bilgileri hırsızlığa karşı korumanıza yardımcı olacaktır.

HTTPS, aşağıdaki özellikleri sağlar:

- Veri Şifreleme: Müdahaleden kaçınmanıza izin verir.
- Veri Güvenliği: Herhangi bir veri değişikliği kaydedilir.
- Kimlik Doğrulama: Kullanıcı yeniden yönlendirmesine karşı korur.

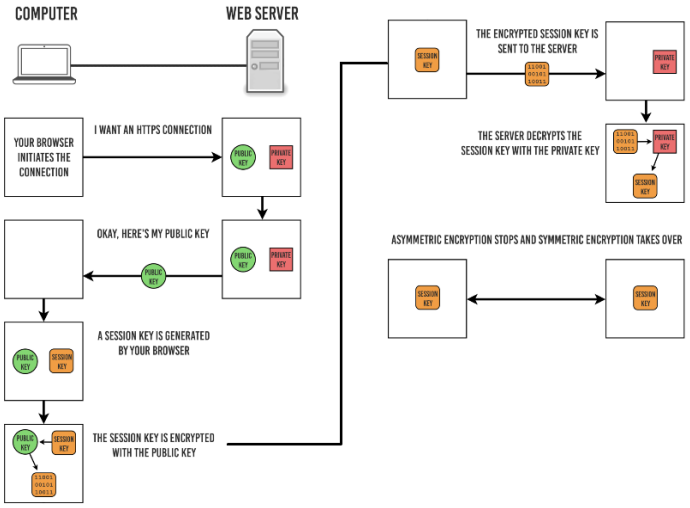
HTTPS, avantajlar:

- Çoğu durumda, HTTPS üzerinden çalışan sitelerin bir yeniden yönlendirmesi olacaktır.
- Protokol, kullanıcıların çevrimiçi bankacılık gibi güvenli e-ticaret işlemlerini gerçekleştirmesine olanak tanır.
- SSL teknolojisi, herhangi bir kullanıcıyı korur ve kaynağa olan güveni artırır.
- Aktarım sırasında bilgiler korur.
- Arama motoru optimizasyonunda avantajlar sağlar.
- Veri bütünlüğü garanti edilebilir

HTTPS, dezavantajlar:

- HTTP ye kıyasla sorgular yavaştır.
- Bilgilerin şifrelenmesi, çözümlenmesi, el sıkışmalar gibi süreçler istemcinin bilgi işlem ve ağ ek yükünü arttırabilir.
- HTTPS, tarayıcıda önbelleğe alınan sayfalardan gizli bilgilerin çalınmasını durduramaz.
- SSL verileri yalnızca ağ üzerinden aktarım sırasında şifrelenebilir dolayısıyla tarayıcı belleğindeki metni koruyamaz

Çalışma şekli aşağıdaki gibidir:



HTTP ve HTTPS kıyaslaması:

HTTP	HTTPS
Etki alanının doğrulanmasını gerektirmez.	Etki alanlarının doğrulanmasını gerektirir.
İletişim için 80 numaralı portu kullanır.	İletişim için 443 numaralı portu kullanır.
Veriler açık metin olarak gönderilir, dolayısıyla iletilen veriler korunmaz.	İletilen verileri şifreleyerek bilgilerin gizliliğini sağlar.
Sertifika doğrulamaları gerektirmez.	SSL Serfikası'nın yetkili tarafından imzalanması ve uygulanması gereklidir.
URL, "http: //" ile başlar.	URL, "https: //" ile başlar.
Güvenli protokol değildir.	Güvenli bir protokoldür.

2- PGP:

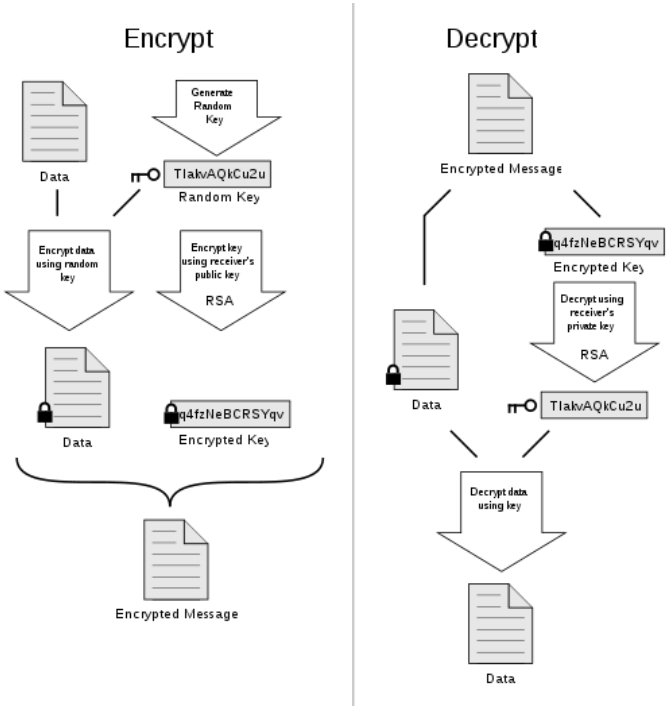
Pretty Good Privacy (PGP), e-posta güvenliği amacıyla tasarlanmış açık kaynaklı bir yazılım paketidir. Phil Zimmerman geliştirdi. Veri şifrelemek, şifreli veriyi çözmek veya veriyi imzalamak için kullanılan, gönderilen ya da alınan verinin gizliliğini ve kimlik doğrulamasını sağlayan bir bilgisayar protokoldür. Genellikle text dokümanlarını, e-postaları, dosyaları, klasörleri ve disk bölümlerini şifrelemek ve imzalamak için kullanılır.

PGP ile şifreleme genel olarak kriptografik özet fonksiyonu, veri sıkıştırma, simetrik anahtar algoritmaları ve açık anahtar şifrelemenin kombinasyonundan oluşan hibrit bir yapıya sahiptir. Bu yöntemle e-posta alıcının posta kutusuna gelene kadar şifrelidir ve içeriği yetkisiz kişilere karşı korunmuş olur.

PGP, aşağıdaki özellikleri sağlar:

- Gizlilik
- Kimlik Doğrulama
- Sıkıştırma
- E-posta Uyumluluğu
- Segmentasyon

Çalışma şekli aşağıdaki gibidir:



3- S/MIME:

Multipurpose Internet Mail Extensions (Çok amaçlı İnternet Posta Eklentileri); E-posta uygulamaları aracılığıyla gönderilecek olan iletiye çeşitli türdeki içeriği eklemek için kullanılan bir İnternet protokolüdür.

E-Posta uygulamalarına ek olarak web tarayıcıları da çeşitli MIME türlerini desteklemektedir. Bu sayede tarayıcı dosya HTML biçiminde veya gösterebileceği türde bir dosya olup olmadığını algılayarak ne yapması gerektiğini bilmektedir.

S / MIME, Çok Amaçlı İnternet Posta Uzantısının (MIME) güvenliği geliştirilmiş bir sürümüdür.

Güvenli / Çok Amaçlı İnternet Posta Uzantıları (S / MIME) S / MIME, elektronik iletiler için standart güvenlik hizmetinin tümünü sağlayan başka bir uygulama katmanı protokolüdür.

S/MIME, e-postalarınızı istenmeyen erişimden korumak için asimetrik şifrelemeye dayanmaktadır. Kullanıcı, güvenilir bir otorite ile bir genel-özel anahtar çifti alır ve ardından bu anahtarları e-posta uygulamalarıyla uygun şekilde kullanır.

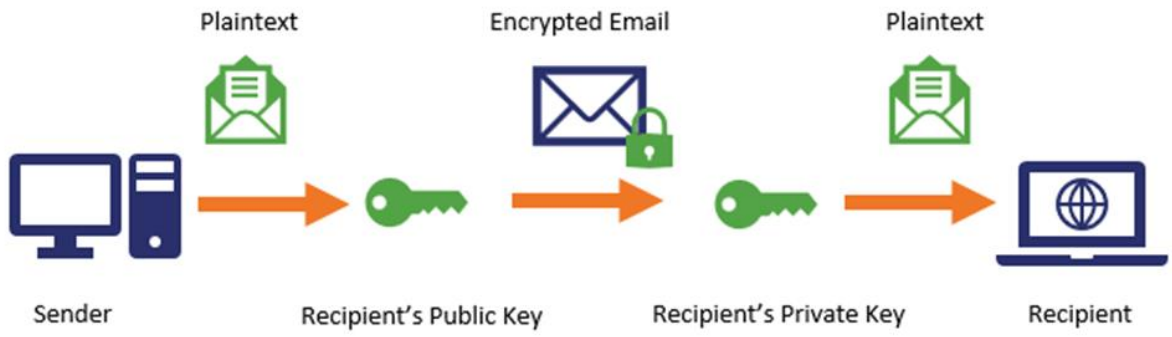
Ayrıca, mesajın meşru göndericisi olarak sizi doğrulamak için e-postalarınızı dijital olarak imzalamanıza olanak tanır ve bu da onu birçok kimlik avı saldırısına karşı etkili bir silah haline getirir.

S/MIME işlevselliği modern e-posta yazılımının çoğuna yerleştirilmiş ve ve bu işlevsellik çalışan yazılımların arasında karşılıklı olarak çalışmaktadır.

S/MIME, elektronik mesajlaşma uygulamaları için aşağıdaki şifreleme güvenlik hizmetlerini sağlar:

- Kimlik doğrulama
- Mesaj bütünlüğü
- İnkar edememezlik
- Gizlilik
- Veri güvenliği

Çalışma şekli aşağıdaki gibidir:



PGP ve S/MIME kıyaslaması:

NUMARA	S/MIME	PGP
1.	PGP'den daha verimlidir.	PGP, S / MIME'den daha az verimlidir.
2.	Tüm uygulamaların güvenli dönüşümü nedeniyle PGP'den daha uygundur.	PGP, nispeten daha az uygundur.
3.	E-postayı ve birçok multimedya dosyasını işlemek için tasarlanmıştır.	Düz metinleri işlemek için tasarlanmıştır.
4.	Yalnızca 1024 genel anahtar içerirken.	PGP, 4096 genel anahtar içerir.
5.	Anahtar değişimi için hiyerarşik olarak geçerli bir sertifikaya dayanır.	Kullanıcı anahtarı değişimine bağlıdır.
6.	S/MIME nispeten pahalıdır.	PGP, S/MIME ile karşılaştırıldığında daha az maliyetlidir.
7.	Elgamal dijital imzasını kullanır.	PGP, Diffie hellman dijital imzasını kullanır .
8.	VPN'lerde kullanılmaz yalnızca e-posta hizmetlerinde kullanılır.	PGP, VPN'lerde de kullanılır.
9.	Endüstriyel kullanım için iyidir.	PGP hem kişisel hem de ofis kullanımı için iyidir.

B- Taşıma Katmanı:

Taşıma katmanı alt katmanlar ve üst katmanlar arasında geçit görevini görür. Alt katmanlar verinin ne olduğuna bakmandan karşı tarafa yollama işini yaparken üst katmanlar da kullanılan donanım ile ilgilenmeden verinin kendisi ile uğraşabilirler.

Bu katmanda üstten gelen veriler parçalara ayrılarak, verilerin bozulması durumunda, hata denetleyici protokolleri devreye alır ve verinin tekrar gönderilmesini sağlar.

Bu şekilde veri kaybı olması durumunda veriler daha küçük boyutlu olacağı için tekrar gönderilmesi daha kolay olur.

Her bir parçaya bir sıra numarası vererek eksik parçaların alıcı tarafından belirlenip tamamlanmasını sağlar.

Bu katmanda TCP, UDP gibi protokoller kullanır.

Kriptografik protokolleri: **SSL – TLS – SSH**

1- SSL:

Güvenli Soket Katmanı (SSL), protokolü internet üzerinden şifrelenmiş güvenli veri iletişimini sağlar.

Özellikle alışveriş sitelerinde, e-posta gönderiminde ve SFTP dosya transferinde güvenlik amacıyla kullanılmaktadır.

SSL, bir web sunucusu ile tarayıcı arasındaki bağlantıyı şifreleyerek aralarında geçen tüm verilerin gizli kalmasını ve saldırılardan uzak kalmasını sağlar.

SSL avantajları:

- Bağlantı gizlidir.
- Gelen verinin kodlanması ve şifreyi çözmesi esnasında güvenlik ve gizliliği sağlar.
- Doküman arşivi oluşturulmasını kolaylaştırır.
- Haberleşen uçların kimlikleri doğrulanabilir.
- Bağlantı güvenilirdir.
- Mesaj akışı, mesajın bütünlüğünün kontrolünü de içerir.
- Veriyi gönderenin ve veriyi alanın doğru yerler olduğunu garanti eder.
- İletilen dokümanların tarih ve zamanını doğrular.

2- TLS:

TLS, bilgisayar ağı üzerinden güvenli haberleşmeyi sağlamak için tasarlanmış kriptolama protokolüdür.

Güvenli Soket Katmanı (SSL) protokolünden türetilmiştir.

TLS, hiçbir üçüncü kişinin herhangi bir mesajı dinlemesini ve değiştirmesini önler.

TLS, avantajlar:

- Güvenli oturum sırasında kullanılan kimlik doğrulama mekanizması, şifreleme algoritmaları ve karma algoritma için işlemler sağlar.
- Şifreleme kullanarak iletilen verilerin güvenliğini sağlamaya yardımcı olur.
- İşlemlerin çoğu istemci tarafından tamamen görünmezdir.
- Çoğu web tarayıcısında ve çoğu işletim sistemi ve web sunucusunda çalışır.

TLS ve SSL kıyaslaması:

TLS	SSL
Fortezza algoritmasını desteklemez.	Fortezza algoritmasını destekler.
Özetlenmiş Mesaj Kimlik Doğrulama Kodu protokolü kullanılır.	MAC protokolü kullanılır.
Daha fazla güvenlidir.	Daha az güvenlidir.
Mesaj özeti, ana sır oluşturmak için kullanılır.	Sözde rasgele işlev, ana sır oluşturmak için kullanılır.
"Sertifika yok" uyarı mesajına sahiptir.	Uyarı mesajını kaldırır ve onu birkaç başka uyarı mesajıyla değiştirir.
Üç SSL sürümü piyasaya sürüldü: SSL 1.0, 2.0 ve 3.0.	TLS'nin dört sürümü yayınlandı: TLS 1.0, 1.1, 1.2 ve 1.3.
SSL'nin tüm sürümleri savunmasız bulundu ve hepsi kullanımdan kaldırıldı.	TLS 1.0 ve 1.1 "bozuldu" ve Mart 2020 itibarıyla kullanımdan kaldırıldı. TLS 1.2, en yaygın kullanılan protokol sürümüdür.

3- SSH:

Güvenli Kabuk (SSH), ağ hizmetlerinin güvenli olmayan bir ağ üzerinde güvenli şekilde çalıştırılması için kullanılan bir kriptografik ağ protokolüdür.

SSH, uzak bilgisayarın kimliğini doğrulamak ve gerektiğinde kullanıcının kimliğini doğrulamasına izin vermek için açık anahtarlı şifreleme kullanır.

En iyi bilinen örnek uygulaması bilgisayar sistemlerine uzaktan oturum açmak için olandır.

SSH, bir SSH istemcisini bir SSH sunucusuna bağlayarak istemci-sunucu mimarisi çerçevesinde güvenli olmayan bir ağ üzerinde güvenli kanal sağlar.

Protokolün en görünür uygulaması, Unıx benzeri işletim sistemlerinde kabuk hesaplarına erişim içindir.

Şifreleme ve şifre çözme gerçekleştirmek için asimetrik şifre kullanır. Birçok şifreleme yöntemi vardır: RSA, DSA, ED25519 vb.

Her zaman bir anahtar çifti halinde çalışır. Açık ve Kapalı anahtar.

Anahtar çiftleri aşağıdaki türlerde olabilir: Kullanıcı, Ana Bilgisayar, Oturum anahtarları.

SSH, avantajları:

- İletim sırasında şifreleme
- Kaynağın kimlik doğrulaması
- Farklı anahtar çiftleri sunar.
- Komut satırı ile denetimi kolaylaştırır.
- Verimli yedeklemeler sunar.

SSH, dezavantajları:

- Yönetimi karmaşıktır.
- DNS mevcut değilse veya kaynak IP adresi çözülmezse kimlik doğrulama gecikmeye sebep olur.
- Daha fazla teknik bilgi gerektirir.
- Çok yaygın değildir.
- Doğrudan bir GUI ye sahip değildir.

SSL ve SSH kıyaslaması:

SSL	**SSH**
443 numaralı port üzerinde çalışır.	22 numaralı port üzerinde çalışır.
Sertifikalara bağlı olduğu için asenkrondur.	Tamamen ağ tüneline bağlıdır.
Kredi kartları ve bankacılık gibi kritik verilerin güvenli bir şekilde aktarılması için en uygun yöntemdir.	İnternet üzerinden komutları güvenli bir şekilde yürütmek için uygun ve etkilidir.
Veri gizliliği sağlamak için hem simetrik hem de asimetrik şifreleme algoritmalarının bir kombinasyonunu kullanır.	Simetrik anahtar algoritmaları kullanarak veri gizliliği sağlar.

C- Ağ Katmanı:

Ağ katmanı, veri paketinin farklı bir ağa gönderilmesi gerektiğinde, veri paketine yönlendiricilerin kullanacağı bilginin eklendiği katmandır.

Bu katmanda, mantıksal adresleme yapılarak, paketlerin protokoller kullanılarak hedefe yönlendirilmesi sağlanır. Böylece cihaz adreslemelerini yönetmiş olur.

Örneğin IP, ARP, ICMP iletişim protokolleri bu katmanda görev yapar.

Ağdaki cihazların lokasyonunu izler.

Verinin taşınması için en iyi yolu belirler.

Bu katmanda harekete geçen datanın hedefe ulaşabilmesi için en iyi yol seçimi yapılır.

7. katmanda oluşturulan her katmanda farklı işlemlerden geçen veri 4. katman olan taşıma katmanında segment haline gelir ve ağ katmanına aktarılır. Bu katmanda adresleme, enkapsülasyon, yönlendirme ve dekapsülasyon işlemleri uygulanır.

Kriptografik protokolleri: **IPSec**

1- IPSec:

İnternet Protokolü Güvenliği(IPsec), Internet Protokolü(IP) kullanılarak sağlanan iletişimlerde her paket için doğrulama ve şifreleme kullanarak koruma sağlayan bir protokol paketidir.

IPsec, içinde bulundurduğu protokoller sayesinde, oturum başlarken karşılıklı doğrulama ve oturum sırasında anahtar değişimlerini gerçekleştirme yetkisine sahiptir.

Oldukça karmaşık bir mekanizmadır, çünkü belirli bir şifreleme algoritması ve kimlik doğrulama işlevinin açık bir tanımını vermek yerine, iletişimin her iki ucunun üzerinde anlaştığı herhangi bir şeyin uygulanmasına izin veren bir çerçeve sağlar.

IPsec kriptografik güvenlik servislerini kullanarak IP protokolü ile gerçekleştirilen bağlantıları korumak için kullanılır.

Ağ seviyesinde doğrulama, veri kaynağı doğrulama, veri bütünlüğü, şifreleme ve replay saldırılarına karşı koruma görevlerini üstlenir.

IPsec'in Özellikleri:

- IPsec, uygulama sürecinden uygulama sürecine değil, bir ağ varlığından başka bir ağ varlığına çalışır. Bu nedenle, bireysel kullanıcı uygulamalarında değişiklik yapılmasına gerek kalmadan güvenlik benimsenebilir.
- Port numarasını taşıyan daha yüksek katman başlıkları gizlendiğinden trafik analizi daha zordur.
- IPsec, aktarım protokolü TCP, UDP, ICMP, OSPF vb. Gibi IP'nin üzerindeki protokollerle çalışır.
- Ağ varlıkları arasında güvenli iletişim sağlamak için yaygın olarak kullanılan IPsec, ana bilgisayardan ana bilgisayara güvenlik de sağlayabilir.
- IPsec, daha yüksek katman başlıkları dahil olmak üzere IP katmanına sunulan tüm paketi korur.
- IPsec'in en yaygın kullanımı, iki konum arasında veya uzak bir kullanıcı ile bir kurumsal ağ arasında bir Sanal Özel Ağ (VPN) sağlamaktır.

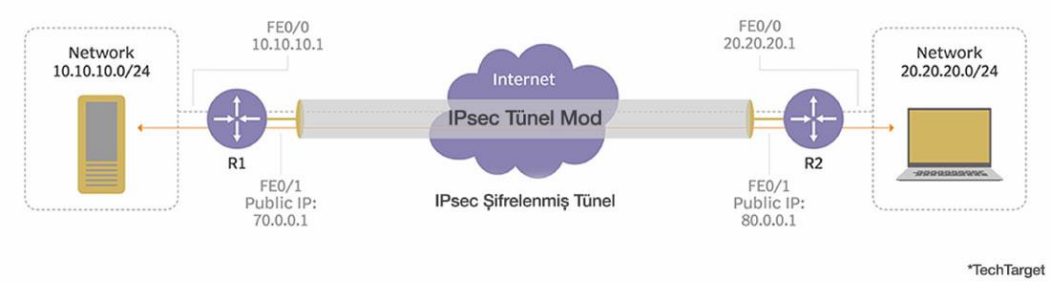
IPsec tarafından sağlanan önemli güvenlik işlevleri aşağıdaki gibidir:

- Gizlilik:
    - İletileri şifrelemek için iletişim düğümlerini etkinleştirir.
    - Üçüncü şahısların gizlice dinlemesini önler.
- Kaynak kimlik doğrulaması ve veri bütünlüğü:
    - Alınan bir paketin, paket başlığında kaynak olarak tanımlanan taraf tarafından gerçekten iletildiğine dair güvence sağlar.

- Paketin değiştirilmediğini veya başka bir şekilde değiştirilmediğini onaylar.

- Anahtar yönetimi:
  - Güvenli anahtar değişimine izin verir.
  - Yeniden oynatma saldırıları gibi belirli güvenlik saldırılarına karşı koruma.

Çalışma şekli aşağıdaki gibidir:

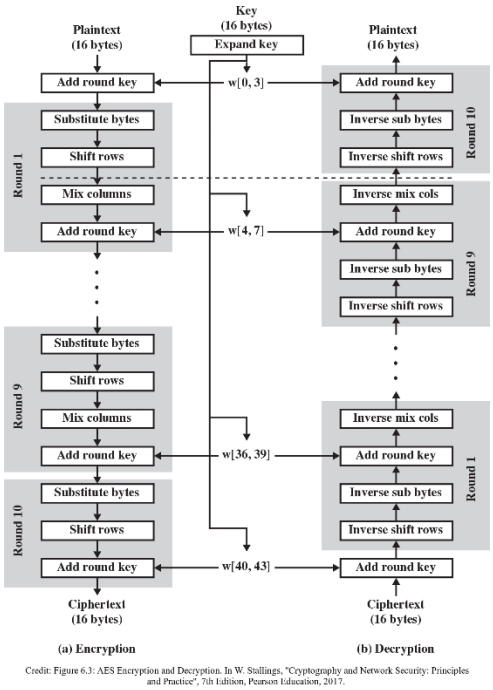


IPSec ve SSL kıyaslaması:

IPSec	SSL
Protokol konfigürasyonu karmaşıktır.	Protokol konfigürasyonu daha basittir.
Güvenlik sağlayan protokol paketidir.	Güvenli bir protokoldür.
Sanal Özel Ağın güvenliğini sağlamak için kullanılır.	Web işlemlerinin güvenliğini sağlamak için kullanılır.
İşletim sistemi alanında bulunur.	Kullanıcı alanında bulunur.
İşletim sisteminde değişiklik gerektirir.	İşletim sisteminde değişiklik gerektirmez.
Kurulum işlemi Satıcıya Özel değildir.	Kurulum süreci Satıcıya Özeldir.

2- KODLAMA KISMI:

A- **AES şifreleme algoritmasının gerçeklenmesi ve şifreleme/deşifrelemede kullanılması(test verileri ile birlikte):**

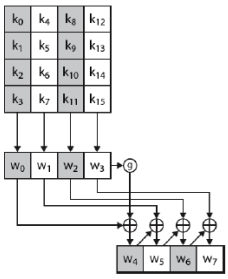


Credit: Figure 6.3: AES Encryption and Decryption. In W. Stallings, "Cryptography and Network Security: Principles and Practice", 7th Edition, Pearson Education, 2017.

AES algoritmasını python programlama dilini kullanarak yazdım. AES adında bir sınıf barındırıyor. Bu sınıfta AES algoritmasındaki prosesleri fonksiyonlar halinde implement ettim. Bu proses aşamaları aşağıdaki gibidir:

- Anahtarı genişletme
- Tur anahtarı ekleme
- Baytları yerine koyma
- Satırları kaydırma
- Sütunları karıştırma

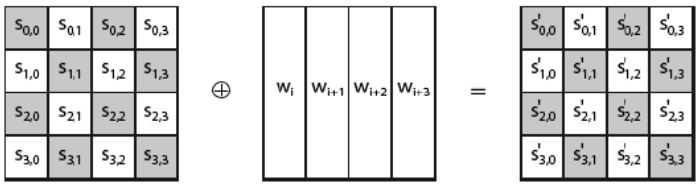
1. **Anahtarı genişletme:**
   Rijndael türü anahtar genişletmesini uygular. Rastgele oluşturulmuş anahtarı, tur fonksiyonunda kullanmak üzere genişletme işlemi gereklidir.
   Daha sonra bu anahtara sırayla döndürme, s-box, xor işlemlerine tabi tutulur.
   Temelde xor işlemi bulunmaktadır, kaynak kodundan ve yorumlarda belirtilmiştir.




```python
# Key schedule core
def core(self, word, iteration):
    # Rotate the 32-bit word 8 bits to the left
    # Key schedule rotate operation.
    # Rotate a word eight bits to the left
    word = word[1:] + word[:1]

    # Apply S-Box substitution on all 4 parts of the 32-bit word
    for i in range(4):
        word[i] = self.getSBox(word[i])

    # XOR the output of the rcon operation with i to the first part (leftmost) only
    # Retrieves a gIVectoren rcon Value
    word[0] = self.xor(word[0], self.rcon[iteration])

    return word

# Key expansion.
def keyExpand(self, key, expandedkeySize):
    size = 16
    currentSize = 0
    iterationRcon = 1
    expandedKey = expandedkeySize * [0]

    # Set the 16 bytes of the expanded key to the input key
    for j in range(size):
        expandedKey[j] = key[j]

    currentSize += size

    while(currentSize < expandedkeySize):
        # Assign the previous 4 bytes to the temporary value temp
        temp = expandedKey[currentSize-4 : currentSize]

        # Every 16 bytes we apply the core schedule to temp and increment iterationRcon afterwards
        if(currentSize % size == 0):
            temp = self.core(temp, iterationRcon)
            iterationRcon += 1

        # We XOR temp with the four-byte block 16 bytes before the new expanded key. This becomes the next four bytes in the expanded key.
        for m in range(4):
            expandedKey[currentSize] = self.xor(expandedKey[currentSize - size], temp[m])
            currentSize += 1

    return expandedKey
```

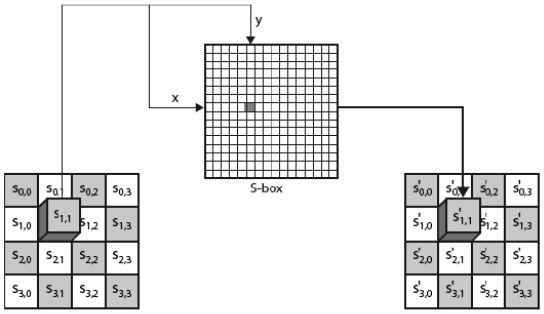
2. **Tur anahtarı ekleme:**

   Tablo ile 128-bitlik tur anahtarını XOR işlemine tabi tutar.



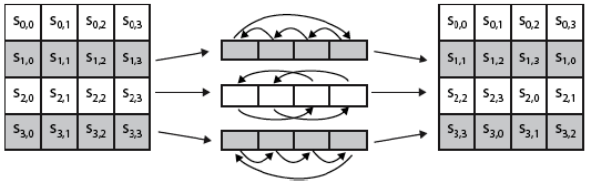
```
# Adds the round key to the st.
def roundKey(self, st, roundKey):
    for i in range(16):
        st[i] = self.xor(st[i], roundKey[i])

    return st
```

3. **Baytları yerine koyma:**
   Önceden belirlenmiş bir tablo ile parametre olarak gelen tablo arası endeks bazlı yerine koyma yapılır.




```
# Substitute all the values from the st with the value in the SBox using the st value as index for the SBox
def bytesSubst(self, st, isInv):
    if (isInv == True):
        getter = self.getInvSbox

    else:
        getter = self.getSBox

    for i in range(16):
        st[i] = getter(st[i])

    return st
```

4. **Satırları kaydırma:**
   Dairesel şekilde bayt kaydırma yapar.
   Satır sayısını endeks olarak alır ve her satır kendi endeksi kadar kaydırılır.
   Şifrelemede sola kaydırma yaparken, deşifreleme de ise sağa kaydırma yapar.



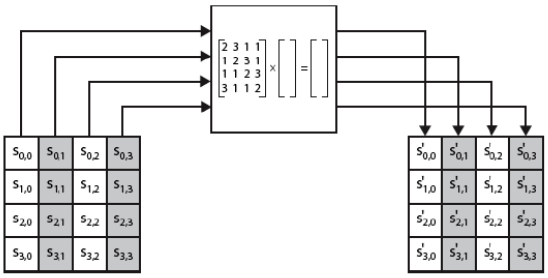
```python
# Iterate over the 4 rows and call shiftRow() with that row
def shiftRows(self, st, isInv):
    for i in range(4):
        stPointer = i * 4
        nbr = i

        # Each iteration shifts the row to the left by 1
        for i in range(nbr):
            if(isInv == True):
                st[stPointer:stPointer+4] = st[stPointer+3:stPointer+4] + st[stPointer:stPointer+3]
            else:
                st[stPointer:stPointer+4] = st[stPointer+1:stPointer+4] + st[stPointer:stPointer+1]

    return st
```

5. **Sütunları karıştırma:**

   4x4 matrisleri ele alır.

   Her sütün farklı şekilde ele alınır.

   Her bayt, sütundaki 4 baytın tümüne bağlı olarak bir değerle değiştirilir.




```python
# Multiplication of 8 bit characters left and right.
def multiple(self, left, right):
    p = 0

    for counter in range(8):

        if(right & 1):
            p = self.xor(p, left)

        hi_bit_set = left & 0x80
        left = left << 1

        # keep a 8 bit
        left = left & 0xFF

        if hi_bit_set:
            left = self.xor(left, 0x1b)

        right = right >> 1

    return p


# Multiplication of the 4x4 matrix
def mixColumns(self, st, isInv):
    # Iterate over the 4 columns
    for i in range(4):
        # Construct one column by slicing over the 4 rows
        column = st[i : i+16 : 4]

        # Apply the mixColumn on one column
        # Multiplication of 1 column of the 4x4 matrix
        if(isInv == True):
            mult = [14, 9, 13, 11]
        else:
            mult = [2, 1, 1, 3]

        cpy = list(column)
        g = self.multiple

        column[0] = g(cpy[0], mult[0]) ^ g(cpy[3], mult[1]) ^ g(cpy[2], mult[2]) ^ g(cpy[1], mult[3])
        column[1] = g(cpy[1], mult[0]) ^ g(cpy[0], mult[1]) ^ g(cpy[3], mult[2]) ^ g(cpy[2], mult[3])
        column[2] = g(cpy[2], mult[0]) ^ g(cpy[1], mult[1]) ^ g(cpy[0], mult[2]) ^ g(cpy[3], mult[3])
        column[3] = g(cpy[3], mult[0]) ^ g(cpy[2], mult[1]) ^ g(cpy[1], mult[2]) ^ g(cpy[0], mult[3])

        # Put the values back into the st
        st[i : i+16 : 4] = column

    return st
```

B- **Gerçeklenen Simetrik şifreleme algoritması kullanılarak CBC ve OFB modlarında çalışmayı gerçekleyip testlerini yapacak şekle getiriniz:**

A partında gerçeklenen simetrik şifreleme algoritması kullanılarak CBC ve CFB modları eklendi. Bu modlar basitçe şu şekildedir.

1. **CBC modu:**
   Mesaj bloklara ayrılır.
   Bir şifreli metin bloğu önceki tüm metin bloklarına bağlıdır çünkü açık metin bir önceki şifreli metin ile xor işlemine tabi tutulur.

$$C_i = AES_{K1}(P_i \text{ XOR } C_{i-1})$$
$$C_{-1} = IV$$

```
elif mode == self.modes["CBC"]:
    for i in range(16):
        if(firstRound == True):
            input[i] =  self.xor(plainText[i], IVector[i])
        else:
            input[i] =  self.xor(plainText[i], cipherText[i])

    firstRound = False
    cipherText = self.encryptn(input, key)

    # Always 16 bytes because of the padding for CBC
    for k in range(16):
        cipherOut.append(cipherText[k])
```

2. **OFB modu:**
   Geri beslemeli bir çıktısı olup mesajı bu çıktı ile XOR işlemine tabi tutar.
   Bu geri besleme mesajdan bağımsızdır ve başlangıç değeri vektördür.

$$C_i = P_i \text{ XOR } O_i$$
$$O_i = AES_{K1}(O_{i-1})$$
$$O_{-1} = IV$$

```
elif mode == self.modes["OFB"]:
    if firstRound:
        output = self.encryptn(IVector, key)
        firstRound = False
    else:
        output = self.encryptn(input, key)

    for i in range(16):
        if(len(plainText)-1 < i):
            cipherText[i] = self.xor(0, output[i])
        elif(len(output)-1 < i):
            cipherText[i] = self.xor(plainText[i], 0)
        elif(len(plainText)-1 < i and len(output) < i):
            cipherText[i] = self.xor(0, 0)
        else:
            cipherText[i] = self.xor(plainText[i], output[i])

    for k in range(end-start):
        cipherOut.append(cipherText[k])
    input = output
```

C- **Herhangi bir doküman üzerinde değişiklik yapılıp yapılmadığını ve yapanın kimliğini anlamak için, özütünü alacak ve sadece işlem yapan kişinin bildiği bir anahtar ile şifreleyip dosyanın sonuna ekleyecek bir araç:**

Temelde bu araç birkaç işlemden oluşmaktadır.

- Girdi olarak aldığı dosyanın içeriğini okuyacaktır.
- Bu içeriği bir özetleme fonksiyonuna tabi tutup, özet değeri elde edilecektir.
- Bu özet değeri oluşturduğumuz AES şifreleme algoritmasına tabi tutulacaktır.
- Bu şifrelenmiş özet değeri dosyanın sonuna yazılacaktır.
- Son durumda dosyanın içeriğinden sonra elde edilmiş 32 baytlık şifreli özet değeri yeni bir dosyaya yazdırılır. Bu dosya sonraki aşamada teyit için kullanılacaktır.

```
# Hash clear text, then encrypt it.
def hash_and_encrypt(cleartext, key):
    hashvalue = hash(cleartext)
    print "Ozet Degeri:", [ord(x) for x in hashvalue], "\n"

    print "Anahtar:", [ord(x) for x in key], "\n"

    cipher = aesmodule.encryptMessage(key, listToString(hashvalue), mode)
    print "Ozetlenmis ve Sifrelenmis Metin:", [ord(x) for x in cipher], "\n"

    return cipher

# In project, Part C
def partC(file, key):
    print "---------- PART C: Ozet Alma / Sifreleme ve Dosyanin Sonuna Ekleme ----------\n"
    cleartext = file.read()
    file.close()

    print "Acik Metin:\n\"" + cleartext + "\"\n"

    cipher = hash_and_encrypt(cleartext, key)

    content = cleartext + cipher
    encryptFilename = "hash_encrypt_" + filename
    encryptFile = open(encryptFilename, "w+")
    encryptFile.write(content)
    encryptFile.close()

    print "Ozet alinidiktan sonra sifrelenen ozet degeri dosyanin sonuna eklenmistir."
    print encryptFilename + " isimli dosya olusturuldu. Son durumu bu dosya icerir"
```

1- Özet fonksiyonu:

Girdi olarak açık metnin uzunluğunu kontrol eder. Daha sonra girdinin uzunluğu 16 nın katı olacak şekilde girdiyi genişletecektir. Bu genişletme işlemi sırasında her bayt için farklı bir karakter kullanılır. ‘a’ karakterinden başlayarak bayt genişletilir. Daha sonra son durumdaki mesaj eşit şekilde sol ve sağ kısımları ayırılır. Bu iki kısım XOR işlemine tabi tutulur. Bu döngü mesaj 16 bayt oluncaya kadar devam eder. Böylece özel bir özetleme fonksiyonu elde edilmiş olur.

```
# Hash message
# First expand to the nearest multiplies of 16
# XOR Left and Rİght sides
# Continiue, until 16 values remains.
def hash(message):
    while(len(message) != 16):
        size = len(message)
        expsize = 16 - (size % 16)

        i = 0
        char = 'a'
        while i < expsize:
            message += char
            char = chr(ord(char) + 1)
            i += 1

        newSize = len(message)
        messageLeft = message[ : newSize//2]
        messageRight = message[newSize//2 : ]
        message = [chr(ord(a) ^ ord(b)) for a,b in zip(messageLeft,messageRight)]
        message = listToString(message)

    return message
```

D- **Dosyanın bütünlüğünün değişip değişmediğinin kontrolü için, c)deki işlemleri yaparak ilk üretilen özüt değer ile karşılaştıran doğrulama aracını gerçekleyerek örnek testleri gösteriniz.:**

Temelde bu araç birkaç işlemden oluşmaktadır.

- Girdi olarak aldığı dosyanın içeriğini okuyacaktır.
- Bu içeriğin son 32 baytı özet değerin şifreli halidir. Bu 32 bayt ayrılır.
- Kalan kısım açık metindir. Açık metin özetleme fonksiyonuna sokulur ve özet değer elde edilir. Bu özet değer doğrulama için kullanılacaktır.
- Elde edilen 32 baytlık şifreli metin şifrelemede kullanılan anahtar ile çözülür. Sonuç olarak şifrelenmeden önceki özet değer elde edilir.
- Son durumda her iki özet değer karşılaştırılır. Aynı ise dosya bütünlüğü korunmuştur, aynı değil ise dosya bütünlüğü bozulmuştur dosyada değişiklik meydana gelmiştir.

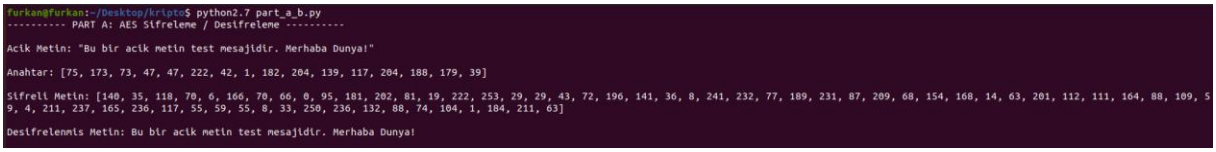
```
# In project prd D
def partD(file, key):
    print "\n---------- PART D: Dosyanin Butunlugunu Dogrulama ----------\n"
    decryptFilename = open(encryptFilename, "r+")
    text = decryptFilename.read()
    decryptFilename.close()


    cleartext = text[:-32]
    print "Gelen Acik Metin:\n\"" + cleartext + "\"\n"

    hashvalue = hash(cleartext)
    print "Gelen Acik Metnin Ozet Degeri:", [ord(x) for x in hashvalue], "\n"

    ency_hash = list(text[-32:])
    print "Gelen Ozeti Alınıp Sifrelenmis Mesaj:", [ord(x) for x in ency_hash], "\n"

    print "Anahtar:", [ord(x) for x in key], "\n"

    decr = aesmodule.decryptMessage(key, ency_hash, mode)
    print 'Gelen Sifrelenmis Ozet Degerin Desifrelenmesi (Cozumlenen Ozet Deger):', [ord(x) for x in decr], "\n"

    print "Gelen Acik Metinin Ozet Degeri ile Gelen Sifrelenmis Ozet Deger Kiyaslandi"

    if(hashvalue == decr):
        print "\n*** Dosya butunlugunun korundugu teyit edilmistir."
    else:
        print "\n*** Dosya butunlugunun korunmadigi tespit edilmistir. Dosyada bir degisiklik meydana gelmistir. "
```

3- PROGRAM ÇIKTILARI VE TEST ÖRNEKLERİ:

A- PART A VE B:

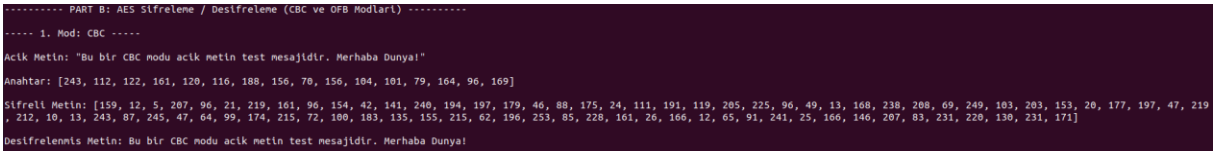
" python2.7 part_a_b.py" yazarak çalıştırınız.
Ekranda şifrelenecek olan açık metin yazdırılır.
Rastgele şekilde oluşturulmuş anahtar yazdırılır.
Bu anahtar kullanılarak açık metin şifreleme fonksiyonu ile şifrelenir ve şifrelenmiş metin ekrana yazılır.
Bu anahtar kullanılarak deşifreleme fonksiyonunda şifreli metin çözülür ve ekrana yazılır.
Aynı şekilde CBC ve OFB için de test edilmiştir.

1- PART A (AES ŞİFRELEME VE DEŞİFRELEME)

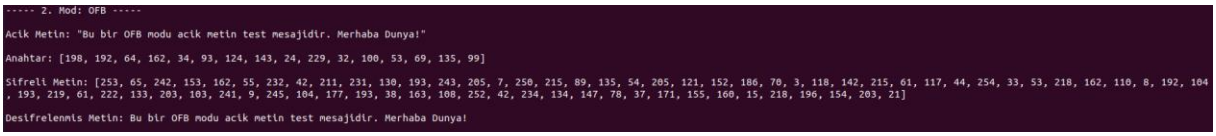


2- PART B (AES ŞİFRELEME VE DEŞİFRELEME CBC VE OFB MODLARI İLE)

CBC:



OFB:

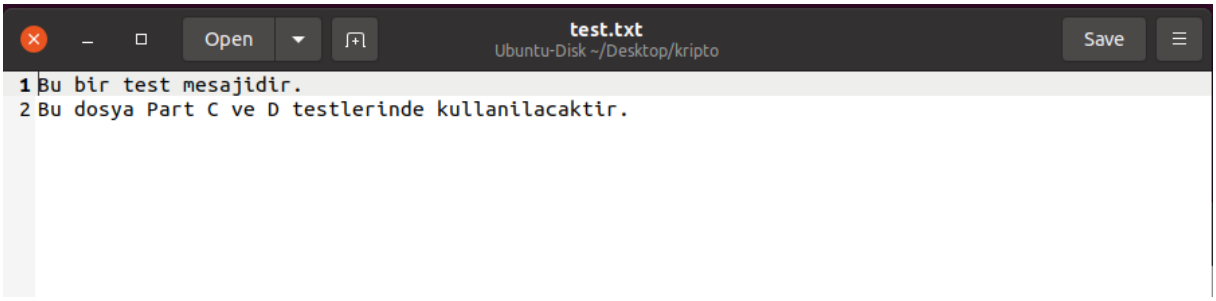


B- PART C VE D:

" python2.7 part_c_d.py test.txt" yazarak çalıştırınız.

Girdi olarak bir dosya alır.

Dosya içeriği:



1- PART C (ÖZET ALMA / ŞİFRELEME VE DOSYANIN SONUNA EKLEME ARACI)

Dosyanın içeriği ekrana yazdırılır.

Bu içerik özet fonksiyonuna gönderilir ve elde edilen özet değer ekrana yazdırılır.

Rastgele oluşturulmuş anahtar ekrana yazdırılır.

Özet değer bu anahtar kullanılarak şifreleme fonksiyonunda şifrelenir, elde edilmiş şifreli özet değer ekrana yazdırılır.

Dosya içeriğinin ardından şifreli özet değer gelecek şekilde "hash_encrypt_test.txt" dosyası oluşturulur. Son durumu bu dosya içerir.

```
furkan@furkan:~/Desktop/kripto$ python2.7 part_c_d.py test.txt
-------------------- TEST 1: DOSYA BUTUNLUGU KORUNUR --------------------
---------- PART C: Ozet Alma / Sifreleme ve Dosyanin Sonuna Ekleme ----------

Acik Metin:
"Bu bir test mesajidir.
Bu dosya Part C ve D testlerinde kullanilacaktir."

Ozet Degeri: [83, 104, 10, 39, 108, 33, 68, 72, 89, 119, 116, 61, 35, 66, 54, 55]

Anahtar: [93, 236, 72, 206, 171, 239, 186, 226, 37, 66, 17, 115, 31, 142, 128, 69]

Ozetlenmis ve Sifrelenmis Metin: [23, 76, 182, 12, 242, 155, 142, 69, 212, 203, 89, 58, 137, 215, 11, 182, 15, 208, 17, 36, 145, 133, 235, 223, 147, 5, 155, 143, 139, 173, 187, 163]

Ozet alinidiktan sonra sifrelenen ozet degeri dosyanin sonuna eklenmistir.
hash_encrypt_test.txt isimli dosya olusturuldu. Son durumu bu dosya icerir
```

2- PART D (DOSYANIN BÜTÜNLÜĞÜNÜ DOĞRULAMA)

Dosyanın içeriğinden son 32 bayt ayrılır. Geri kalan açık metindir. Açık metin ekrana yazdırılır.

Gelen açık metnin özet değeri özetleme fonksiyonundan elde edilir ve ekrana yazdırılır.

Ayrılan 32 bayt şifrelenmiş özet değerdir. Bu mesaj ekrana yazdırılır.

Şifrelemede kullanılan anahtar ekrana yazdırılır.

Gelen şifrelenmiş özet değer, anahtar ile deşifreleme fonksiyonuna gönderilir ve şifrelenmeden önceki özet değer elde edilmiş olur. Bu özet değer ekrana yazdırılır.

Daha sonra gelen metnin özet değeri ile şifresi çözülmüş özet değer kıyaslanır. Aynı ise dosya bütünlüğü korunmuştur mesajı yazdırılır, aynı değil ise dosya bütünlüğü bozulmuştur mesajı yazdırılır.

```
---------- PART D: Dosyanin Butunlugunu Dogrulama ----------

Gelen Acik Metin:
"Bu bir test mesajidir.
Bu dosya Part C ve D testlerinde kullanilacaktir."

Gelen Acik Metnin Ozet Degeri: [83, 104, 10, 39, 108, 33, 68, 72, 89, 119, 116, 61, 35, 66, 54, 55]

Gelen Ozeti Alınıp Sifrelenmis Mesaj: [23, 76, 182, 12, 242, 155, 142, 69, 212, 203, 89, 58, 137, 215, 11, 182, 15, 208, 17, 36, 145, 133, 235, 223, 147, 5, 155, 143, 139, 173, 187, 163]

Anahtar: [93, 236, 72, 206, 171, 239, 186, 226, 37, 66, 17, 115, 31, 142, 128, 69]

Gelen Sifrelenmis Ozet Degerin Desifrelenmesi (Cozumlenen Ozet Deger): [83, 104, 10, 39, 108, 33, 68, 72, 89, 119, 116, 61, 35, 66, 54, 55]

Gelen Acik Metinin Ozet Degeri ile Gelen Sifrelenmis Ozet Deger Kiyaslandi

*** Dosya butunlugunun korundugu teyit edilmistir.
```

3- PART C-D (DEĞİŞTİRİLMİŞ DOSYA İÇERİĞİ İLE TEST2)

```
-------------------- TEST 2: DOSYA BUTUNLUGU KORUNMAZ --------------------
---------- PART C: Ozet Alma / Sifreleme ve Dosyanin Sonuna Ekleme ----------

Acik Metin:
"Bu bir test mesajidir.
Bu dosya Part C ve D testlerinde kullanilacaktir."

Ozet Degeri: [83, 104, 10, 39, 108, 33, 68, 72, 89, 119, 116, 61, 35, 66, 54, 55]

Anahtar: [93, 236, 72, 206, 171, 239, 186, 226, 37, 66, 17, 115, 31, 142, 128, 69]

Ozetlenmis ve Sifrelenmis Metin: [166, 140, 134, 70, 231, 182, 33, 36, 75, 0, 23, 61, 238, 228, 99, 7, 225, 107, 179, 172, 80, 23, 70, 114, 182, 116, 2, 22, 164, 115, 127, 111]

Ozet alinidiktan sonra sifrelenen ozet degeri dosyanin sonuna eklenmistir.
hash_encrypt_test.txt isimli dosya olusturuldu. Son durumu bu dosya icerir
```

```
---------- PART D: Dosyanin Butunlugunu Dogrulama ----------

Gelen Acik Metin:
"change file content r.
Bu dosya Part C ve D testlerinde kullanilacaktir.]fËH`xó›B◆T"

Gelen Acik Metnin Ozet Degeri: [166, 70, 114, 14, 51, 96, 55, 182, 5, 90, 41, 9, 16, 3, 28, 5]

Gelen Ozeti Alınıp Sifrelenmis Mesaj: [121, 25, 194, 184, 195, 186, 195, 164, 197, 189, 226, 128, 156, 76, 226, 128, 157, 195, 183, 55, 18, 195, 152, 19, 195, 162, 194, 173, 97, 77, 108, 81]

Anahtar: [93, 236, 72, 206, 171, 239, 186, 226, 37, 66, 17, 115, 31, 142, 128, 69]

Gelen Sifrelenmis Ozet Degerin Desifrelenmesi (Cozumlenen Ozet Deger): [45, 83, 214, 239, 106, 130, 71, 238, 110, 253, 3, 84, 155, 116, 188, 204]

Gelen Acik Metinin Ozet Degeri ile Gelen Sifrelenmis Ozet Deger Kiyaslandi

*** Dosya butunlugunun korunmadigi tespit edilmistir. Dosyada bir degisiklik meydana gelmistir.
```