Co-funded by the Erasmus+ Programme of the European Union

# InCyT Eğitim Modülü

| | |
|---|---|
| **Çalışanlar için Bilgi Güvenliği<br>Ünite 2 Hafta 3<br>Birim adı: Kriptografiye Giriş** | |
| ***Öğrenme Aktiviteleri*** | ❑ **Web Semineri ❑Alıştırmalar ❑Atölye Çalışması ❑Sunum ❑Diğer** |
| ***Öğrenme Sorumlusu*** | |
| ***Öğrenme Etkinliği Hedefleri*** | 1. Şifreleme<br>2. Kriptoanaliz<br>3. Tek seferlik ped algoritması<br>4. Simetrik algoritmalar<br>5. ymmetrik algoritmalar olarak<br>6. Hash fonksiyonus<br>7. İmzalar ve ortak anahtar altyapıları<br>8. Kriptografik Jetonlar<br>9. Meydan okuma-yanıt mekanizması |
| ***Kazanılacak Beceriler*** | • Kriptografi algoritmaları ve altyapıları hakkında temel bilgiler<br>Teknik Beceri 2 - TS2 |
| ***Öğrenme aktivitesinin süresi*** | 1 hafta |
| ***Öğrenme rdonatımları*** | ***Neye ihtiyaç var?***<br>• Önceden özel bilgi gerekmez |

Co-funded by the
Erasmus+ Programme
of the European Union

## Modül hakkında bilgi

Kriptografi, hem iletişim sırasında hem de arşivleme amacıyla bilgilerin açıklanmasını önlemek için kullanılan metodolojilerle ilgilidir. Hedeflerine, yetkisiz bir üçüncü tarafın bilgileri almasını önleyen protokollerin tasarlanması ve uygulanması ile ulaşılmaktadır. Güvenli bir iletişimde, düşman üçüncü taraf iletilen bilgilere erişemez, arşivleme durumunda üçüncü taraf arşivlenenlerin içeriğini izleyemez.

Kriptografide, üçüncü taraf her zaman gizlilik, ticari sır veya ulusal güvenlik amacıyla erişmemesi gereken bilgileri almayı amaçlayan kötü niyetli bir varlıktır. Veri gizliliği ve bütünlüğü, ilgili tarafların kimlik doğrulaması ve veri kaynağının ilettiği veya sakladığı şeylerin reddedilmemesi, modern kriptografinin temel ilkeleridir.

Temel kavramların açıklandığı giriş bölümünden sonra, *One Time Pad Algoritmasından* başlayarak ve daha sonra iki şifreleme algoritma ailesini göz önünde bulundurmak için temel şifreleme tekniklerinin bir ilerlemesini sunuyoruz: Simetrik Algoritmalar ve Asimetrik (veya Açık Anahtar) Algoritmalar, bunların karşılaştırılmasıyla. Dijital İmzaları tanıtmak için Karma İşlevlerin rolü de tartışılmaktadır. Ortak Anahtar Altyapılarının anahtar kavramının bir açıklaması, standart mekanizmalar ve biçimler hakkında bazı ayrıntılarla derinlemesine sunulmaktadır. Sonuç olarak, Kriptografik Belirteçler ve Meydan Okuma-Yanıt Mekanizması hakkında bazı bilgiler sağlanmaktadır.

## Öğrenme Materyalinin İçeriği

## Giriş

**Kriptografi** , bir iletinin içeriğinin diğer varlıklardan gizlendiği mekanizmadır. Başka bir deyişle, şifreleme, mesajları şifrelemekle, yani düz metin almakla, bir algoritma ve genellikle bir anahtar kullanarak, mesajın içeriğini anlaşılmaz hale getirmekle ilgilenir.

**Kriptanaliz** , anahtarı bilmeden mesajın şifresinin nasıl çözüleceğini inceler. Bir saldırıda, kriptanaliz, anahtara sahip olmadan iletinin içeriğini izlemeye çalışmaktan ibarettir. Kriptograflar ve kriptanalistler aslında aynı insanlardır: kriptografik algoritmaları yazan kişi, zayıf yönlerini bulmak veya sağlamlıklarını doğrulamak için onları analiz eder. Her zaman olduğu gibi, güvenlikle uğraşırken, saldırı tekniklerinin nasıl çalıştığını bilmeden koruma sorunlarıyla başa çıkabilme fikri çok etkili değildir.

Mesajlardaki bilgiler genellikle gereksizdir, resmi anlamda, mesajlar genellikle bilgiyi taşımak için gerçekten gerekli olandan çok daha fazla sayıda bit kullanır. **Bilginin fazlalığı kriptanalizi kolaylaştırır**: bilgilerdeki fazlalık, mesajların mesajın içeriğini izlemeyi kolaylaştıran bir yapı gösterebileceği anlamına gelir, çünkü tüm mesajlar eşit derecede olası değildir. Bayt cinsinden normal ASCII karakterleriyle temsil edilen bir metin düşünürsek, anlamlı iletileri temsil edebilecek bit kombinasyonlarının tüm olası kombinasyonlara kıyasla nispeten az olduğunu biliriz. Belirli sayıda baytla temsil edilebilen gerçek mesajlar çok azdır. Bu, doğru bilgi taşımayanları atarak geçerli iletileri tanımlamayı mümkün kılar. Bütün bunlar kriptanalistin faaliyetine büyük ölçüde yardımcı olur, bu nedenle şifreleme algoritmasının amacı da bilgileri daha iyi gizlemek için bu fazlalığı gizlemektir.

**Örnek:** MAVİ, YEŞİL, KIRMIZI ve SARI olmak üzere dört renkten oluşan *bir diziyi* kodlama.

Bunlar kodlanabilir:

- 2 bit ile (00, 01, 10, 11)
- 4 farklı büyük harf olarak
- renk adlarını şifreleme

İlk durumda, minimum varlık sayısı (bit) kullanılır, her renk adını karşılık gelen 2 bit ile değiştirerek orijinal sırayı gizleyecek, ancak COLOR ile 2 bit arasında bir yazışma olduğu için sırayı gizlemeyecektir. İkinci durumda, her harf ASCII kodunda 8 bit kullandığından, her renk için 6 bit gereksizdir → % 75 yedeklilik ve 2 bit ile aynı gizleme derecesini elde ederiz. Renkleri ve değerleri hala ilişkilendiremiyoruz.

Aşağıdaki çok basit şifreyi, alfabedeki harfleri belirli sayıda pozisyona göre kaydırmaktan başka bir şey yapmayan Sezar şifresi olarak adlandırılan şifreyi düşünün. Örneğin, onları 3'e kaydırırsak, tüm 'A'lar 'D' olur, tüm 'B'ler 'E' olur vb. Bu çok basit bir şifredir; aslındaJül Sezar'a attr, ancak renk adları bu yöntemle şifrelenirse, kodlanmış harflerinin tekrarlanan deseni kolayca tanımlanır (R için

ASCII kodu, ardından E'nin ASCII kodu ve ardından D'nin ASCII kodu bir varlık olarak tanınması kolaydır). Böylece dizi tanımlanabilir ve eğer kelimeler de biliniyorsa, bu durumda sadece harf sayısını saymak orijinal kelimeleri tanımlamak için yeterlidir.

Bu basit örnek, kriptanalizin mesajın içeriğinden bağımsız bir analiz olmadığını anlamamızı sağlar: tamamen rastgele bir mesajın tanımlanması ve analiz edilmesi, tanınabilir bir yapıya sahip bir mesajdan çok daha zor olacaktır.

Bazı şifreleme saldırıları ve algoritma türleri, metnin bir bölümünü bilmeye dayanır. Bazı durumlarda, metnin bir bölümünü bilmek, anahtarı izlemeyi ve ardından metnin geri kalanına erişmeyi mümkün kılar. Ağ iletişimi söz konusu olduğunda bu çok önemlidir, çünkü ağ iletişimlerinde farklı seviyelerdeki protokoller genellikle her durumda tanınabilir olan belirli miktarda bilgiye sahiptir, örneğin bir IP paketinde başlığın gönderen adresini ve alıcı adresini içerdiğini biliyoruz.

Bir şifreleme algoritması gereksiz bilgileri de gizler, bu nedenle şifrelenmiş bir mesaja sıkıştırma algoritması uygulamak çok fazla avantaj sağlamaz, çünkü sıkıştırma oranını azaltır. Sıkıştırma ve şifreleme kullanmak istiyorsak, önce sıkıştırma ve ardından şifreleme kullanmalıyız.

## Tek Seferlik Ped Algoritması

One Time Pad (OTP), tabiri caizse "mükemmel" bir algoritmadır. Çok basit bir şifreleme algoritmasıdır, uygulanması çok basittir ve onu özellikle ilginç kılan bir özelliğe sahiptir: anahtarı bilmeden orijinal metne geri dönmenin imkansız olduğu kanıtlanmıştır. Şimdi ona bakıyoruz ve sonra neden bu kadar iyi bir algoritmaya sahip olduğumuza göre, başkalarını bulmamız gerektiğini tartışıyoruz.

EXCLUSIVE OR ⊕ (XOR), iki işlenen farklı olduğunda sonucun Doğru olduğu, mantıksal değerler üzerinde çalıştığı, ancak genellikle Doğru = 1 ve Yanlış = 0 olmak üzere ikili basamaklara çevrildiği mantıksal bir işlevdir, *doğruluk tablosu* şöyledir:

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

Bu tablodan, $X \oplus Y \rightarrow Z$ ise, $Z \oplus Y \rightarrow X$ ve $Z \oplus X \rightarrow Y$ olduğu açıktır. EXCLUSIVE OR çok basit bir işlevdir ve elektronik devre olarak uygulanması kolaydır.

OTP mekanizması iki taraftan bir mesaj aktarmak için kullanılır. Bilimsel literatürde, aktörleri kişisel isimlerle tanımlamak yaygın bir uygulamadır, genellikle Alice (A) ve Bob (B) olarak adlandırılırlar. Bu yüzden Alice'in *Bob'a* n-baytlık bir mesajı (*PlainText*) nasıl güvenli bir şekilde iletebileceğini açıklayacağız.

One Time Pad algoritması bir kurulum aşaması gerektirir:

- Tam olarak *n* baytlık rastgele bir dizi oluşturulur, bu *One Time Pad* (OTP) veya sadece Anahtardır;
- OTP, iki parça arasında bir şekilde güvenli bir şekilde paylaşılır (önceden verilir, zaten güvenli bir kanalda gönderilir, vb.).

Her iki taraf da anahtara sahip olduğunda, mesajları iletmek için kullanabilirler. Alice'in Bob'a bir mesaj göndermek istemesi durumunda:

- Alice, Anahtarla Düz Metin'in ÖZEL VEYA ÖZEL VEYA bir *CypherText* üreterek *azar azar* hesaplar:
  **Şifreleme:** Düz Metin ⊕ Anahtar → CypherText
- Ortaya çıkan *CypherText* (Şifreli Metin), Alice tarafından Bob'a muhtemelen güvensiz bir kanal üzerinden iletilir.
- Bob, *CypherText* ve *Key'in* EXCLUSIVE VEYA değerini azar *azar* hesaplar, böylece orijinal PlainText'i kurtarır:
  **Şifre çözme:** CypherText ⊕ Anahtar → Düz Metin

**Örnek**

Hem Alice hem de Bob'un rastgele bir OTP (*Anahtar*) üzerinde anlaştıklarını varsayalım.

**Alice,** Bob'a "HELLO" göndermek istiyor (ASCII Kodunda, okuma kolaylığı için boşluklar burada tanıtılmıştır):

**H E L L O**

*Düz Metin*: 01001000 01000101 01001100 01001100 01001111

⊕

*Anahtar*: 10101001 01110010 10110010 10000110 11000110

=

*CypherText*:11100001 00110111 11111110 11001010 10001001

**Bob,** *CypherText'i* alır ve Anahtarı uygular:

*CypherText*:11100001 00110111 11111110 11001010 10001001

⊕

*Anahtar*: 10101001 01110010 10110010 10000110 11000110

=

*Düz Metin*: 01001000 01000101 01001100 01001100 01001111

**H E L L O**

*Düz Metin* geri çevrilmiştir.

*Anahtar* rastgele bir dizi olduğundan, dizinin belirli bir bitinin '0' veya '1' olup olmadığını söylemenin bir yolu yoktur, çünkü eşit derecede olasıdırlar. Anahtar olmadan, bir *CypherText* biti verildiğinde, orijinal bitin ne olabileceğini tahmin etmeye çalışma şansımız yoktur. Bu, tek bitler için, mesajın tamamı için, mesajımızın herhangi bir parçası için geçerlidir.

Anahtarın bir kısmının bilinmesi durumunda, Düz Metin'deki ilgili kısmı kurtarmak mümkündür, ancak Anahtarın geri kalanı olmadan *Düz* Metin'in diğer bitleri hakkında bilgi eksikliği yoktur. Bu nedenle, daha önce bahsedilen kriptanaliz etkili değildir. Bu durumda, özgün iletinin kalan kısmını bulmak için, Anahtarın kalan kısmı gereklidir. Anahtarın ilettiği tüm bilgiler, metnin tüm bilgilerini kurtarmak için gereklidir.

## Tek Kullanımlık Ped Sorunları

OTP mekanizmasının gücü, *Anahtarın* uzunluğunun metnin uzunluğuna tam olarak eşit olmasına ve yalnızca bir kez kullanılmasına bağlıdır. Ana dezavantajı, OTP'nin iletiminin mesajla aynı sorunlara sahip olmasıdır.

Bazı durumlarda bu sorun kolayca çözülür, örneğin bir elçiliğe çok uzun bir anahtar içeren bir şerit içeren diplomatik bir evrak çantası sağlanabilir ve daha sonra bu elçilikle değiş tokuş edilecek mesajlar, anahtarın bitlerini yavaş yavaş tüketerek şifrelenebilir. Bunlar çok tuhaf kullanımlardır, ancak oldukça etkilidir. Ağ iletişimini göz önünde bulundurursak, One Time Pad'in gerçek kullanımlarının çok sınırlı olduğu açıktır. Daha esnek bir şeye, çok uzun süre kullanılabilecek bir anahtarın değişimine izin veren bir şeye ihtiyaç vardır.

Sorunların her birini ele alalım.

**Önceden çevrimdışı iletim**

Bu, anahtarlarla ilgili genel bir sorundur: anahtarların taraflarca bir şekilde kararlaştırılması gerekir. Bu anahtarlar üzerinde hemfikir olma ihtiyacı, kriptografi kullanmanın en önemli sorunudur. Kriptografi ile ilgili sorun, anahtar yönetiminin yanı sıra çok fazla algoritma ve kriptanaliz değildir: bu, tüm ve yalnızca doğru konuların anahtarlara erişebildiğinden emin olmak, daha sonra bunları önceden değiştirmek ve değiştirilen anahtarların doğru olduğundan emin olmaktır.

**Tek seferlik kullanım**

One Time Pad, adından da anlaşılacağı gibi, yalnızca bir kez kullanılır, bu önemli bir husus ve sınırlamadır. Aynı anahtarı iki mesajda kullanmaya çalışırsak, yöntemin güvenliğini zayıflatırız.

**Deterministik bir sistemde rasgele sayılar üretmek çok zordur**

Genel amaçlı bilgisayarlar, her sayının bir öncekinden bir miktar hesaplama ile hesaplandığı sözde rasgele sayı üretimi kullanır, bu da çok uzun bir süre sonra tekrarlanan bir sayı dizisi üretir. Bunlar gerçek rasgele sayılar değildir, çünkü bir bilgisayar deterministik bir makinedir. Bununla birlikte, kriptografide, bu özellik zararlıdır. Bir öncekinden bir sonraki rasgele sayıyı türetmek mümkünse, şifreleme algoritmaları veya protokolleri tarafından sunulan korumalar çok daha az güvenli hale gelir. Şifreleme algoritmalarını uygulayan kod yazmak önemsiz değildir: dikkat edilmesi gereken çok fazla ayrıntı vardır, kod geliştiricileri tarafından yıllar içinde tespit edilen ve yalnızca bu alandaki kod geliştiricilerinin yönetmeyi öğrendiği tüm kusurlar. Bu, iyi test edilmiş, yaygın olarak kullanılan, kapsamlı olarak doğrulanmış kitaplıklar kullanmak yerine şirket içi şifreleme algoritmaları geliştirmenin ve / veya uygulamanın her zaman kötü bir fikir olduğu anlamına gelir.

## OTP dışındaki algoritmalar

Diğer algoritmalar sabit uzunlukta anahtarlar kullanır. Bir $n$ bit anahtarı kullanırsak, $2^n$ farklı anahtarımız olur, n'nin değeri yeterince büyükse ve sadece kaba kuvvet saldırısı uygularsanız (yani tüm olası anahtarları denerseniz), doğru olanı bulmak çok uzun yıllar alır, böylece "muhtemelen" mesajın içeriği artık ilgi çekici değildir. Örneğin, mevcut en hızlı süper bilgisayarlardan biriyle, 256 bitlik bir anahtarı kontrol etmek $3{,}67\times10^{55}$ yıl gerektirir. Açıkçası, yeterli uzunluktaki anahtarları kullanmak, meşru olarak şifreleyen ve şifresini çözenler için bile bazı ek hesaplama maliyetlerine sahiptir, ancak etki sınırlıdır.

Simetrik ve Asimetrik algoritmalar vardır.

## Simetrik Algoritmalar

En tipik ve eski şifreleme algoritmaları ailesi, mesajları şifrelemeyi düşündüğümüzde tipik olarak düşündüğümüz ***algoritmalar***, şifreleme ve şifre çözme işlemlerinin simetrisinden sonra adlandırılan

simetrik algoritmalardır. Bu algoritmalar, hem gönderen hem de alıcı tarafından bilinmesi gereken tek bir anahtar kullanır; bir iletiyi şifrelemek ve ardından şifresini çözmek için kullanılır.

Basit bir simetrik şema OTP ile aynı olabilir, çünkü OTP simetrik bir algoritmadır, büyük farkla *Anahtarın* sabit bir uzunluğa (örneğin 256 bit) sahip olması ve muhtemelen her seferinde karmaşık bir değişiklikten sonra bir şekilde tekrarlanmasıdır:

**Şifreleme:** *encryption_function* (Düz Metin, Anahtar) → CypherText

**Şifre çözme:** *decryption_function*(CypherText, Key) → Düz Metin

Daha genel olarak, aynı *Anahtarı* kullansak bile , şifresini çözmek için kullanılan işlev, şifrelemek için kullanılanla aynı olmak zorunda değildir. Örneğin, Sezar'ın şifresini göz önünde bulundurursak, mesajı deşifre eden işlevin alfabeyi sağa değil sola kaydırdığı açıktır, ancak şifre çözme aynı anahtarı kullanır: sayı 3.

## Bazı önemli simetrik anahtar algoritmaları

*DES (Veri Şifreleme Standardı)*

Artık modası geçmiş olan bu tarihsel algoritma, 1970'lerin sonlarında (algoritma geliştirildiğinde) birçok kullanım için kesinlikle yeterli olan 56 bitlik bir anahtar kullanıyor, çünkü mesajı o sırada mevcut olan bilgi işlem kaynaklarıyla ve bu algoritmanın kullanıldığı bağlamlarda almak için 56 bitlik bir anahtara kaba kuvvet saldırısı, pratik değildi. 40 yıl sonra DES algoritması bir ev bilgisayarının birkaç dakikasına bile dayanamayacaktı. Bu çok önemli bir husustur: anahtarın uzunluğu, mevcut *gerçek* bilgi işlem yeteneklerine göre seçilir. Şimdi yeterince sağlam olduğu düşünülen bir anahtar, 20 yıl içinde artık o kadar güçlü olmayabilir. Genel olarak, hem bilgi işlem yetenekleri arttığından, hem de bir disiplin olarak kriptanalizin ilerleme kaydettiği ve algoritmadaki zayıflıkları bulabileceğinden bir miktar marj olduğundan emin olmak için anahtarın uzunluğuyla bolca bulunma eğilimindeyiz. Bu oldukça ilginç bir yönüdür. Birinin düşündüğünün aksine, normalde bir şifreleme algoritması aniden başarısız olmaz, yani bir algoritmada bir kusur keşfedildiği gün, anında güvenliden tamamen güvensizliğe dönüşür ve herkes bir anda düz metni bulabilir. Olan şey, araştırmacıların algoritmayı analiz etmeleri ve belki de bazı durumlarda ona saldırmayı kolaylaştırabilecek küçük zayıflıklar bulmalarıdır. Sonra birisi muhtemelen sorunu çözmek için bazı algoritma optimizasyonları bulur. Zaman geçtikçe, çözülemeyen problemler nedeniyle algoritma, belirli kullanımlar için artık yeterli olmayana kadar yavaş yavaş zayıflar. Kriptografik algoritmaların savunmasında ilerici bir erozyon var ve bu ilerici zayıflama yönetilmesi daha zor bir sorundur, çünkü kriptanaliz açısından gelecekte ne gibi bir ilerleme olacağını bilemeyiz.

*AES (Gelişmiş Şifreleme Standardı, orijinal adı Rijndael olarak da bilinir)*

AES sadece sağlamlığa değil, aynı zamanda tipik mimarilerde uygulamanın verimliliğine göre seçildi. Gizli bir algoritma aranmadı, çünkü gizli bir algoritma gizli olmayan bir algoritmadan daha sağlam

olarak kabul edilmez: aksine, algoritmanın özünde sağlam olduğundan emin olmak için mümkün olduğunca çok sayıda değerli kriptanalist tarafından analiz edilmesi amaçlanmıştır. Güvenli ama çok yavaş, bilgi işlem kaynakları açısından çok pahalı bir algoritma kullanılamaz. Bu kısıtlama, sağlamlığa kıyasla, rakiplerin üstesinden gelmesi en zor kısıtlama olarak belirlendi.

## Asimetrik (veya Ortak Anahtar) Algoritmalar

Asimetrik algoritmalar sadece bir anahtar değil, iki anahtar kullanır: bir anahtar şifrelemek ve diğeri şifresini çözmek için kullanılır. Bu nedenle, mesaj alışverişinde, gönderen ve alıcı aynı anahtarı kullanmaz. Bu anahtarlar çiftler halinde oluşturulur ve yönetilir: bir anahtarın (çiftten herhangi birinin) şifrelediği şeyin şifresi yalnızca çiftin diğer anahtarı tarafından çözülebilir.

Asimetrik algoritmaları kullanmanın nedeni, simetrik kriptografinin sahip olduğu sorunlardan kaynaklanmaktadır:

1. 2 varlık arasındaki gizli iletişim, benzersiz bir anahtarın yalnızca aralarında paylaşılmasını gerektirir, eğer varlıklar n ise, o zaman $n\cdot(n-1)/2$ tarafların her bir çifti arasında gizli iletişime izin vermek için farklı anahtarlar gereklidir.
2. Both gönderen ve alıcı paylaşılan anahtarlarını bilir, bu nedenle *reddetmemek* (reddetmek, belirli bir mesajın yazarı olmayı reddetmektir) mümkün değildir.

İlk sorun, her bir öznenin yönetmesi gereken anahtar sayısıyla ilgili değil, sorun onları *değiştiriyor*. Her konu, aslında karmaşık bir işlem olan diğer n-1 konularının her biriyle bir anahtar üzerinde hemfikir olmalıdır; bu, simetrik kriptografiyi birçok kullanım için pratik hale getirmez.

İkinci sorun, *reddetmeme*, genellikle belirli bir mesajın belirli bir konu tarafından gerçekten gönderildiğinden emin olmak istediğimiz gerçeğinde yatmaktadır. Bir konu başka bir konudan bir mesaj alırsa ve iki konu onu şifrelemek için kullanılan anahtarı bilen tek kişiyse, her konunun diğer konunun mesajın yazarı olduğundan emin olduğu açıktır. Bunun tersi de geçerlidir, çünkü yalnızca anahtarı bilirler, her konu yalnızca diğer öznenin mesajı okuyabileceğini bilir. Sorun, bu mekanizma çalışmadığı için bunu üçüncü bir tarafa, bir tür dava için kanıtlamaya ihtiyaç duyulduğunda ortaya çıkar: her taraf bir mesaj oluşturabilir, şifreleyebilir ve ardından diğer tarafın yazar olduğunu söyleyebilir. *Reddetmeme* problemi simetrik kriptografi kullanılarak çözülemez.

Asimetrik şema simetrik şemaya benzer, ancak kullanılan anahtarlar çift olarak oluşturulan ikisidir:

**Şifreleme:** *encryption_function* (*Düz Metin*, ***Key1***) → *CypherText*

**Şifre çözme:** *decryption_function*(*CypherText*, ***Key2***) → *Düz Metin*

İki anahtar matematiksel olarak değiştirilebilir olduğundan, *EncryptionKey* ve *DecryptionKey* yerine ***Key1*** ve ***Key2*** adları kullanılmıştır . Anahtarları üreten özne genellikle bir özel/gizli tutar (özel anahtar olarak adlandırılır) ve diğerini genel yapar ( ortak anahtar olarak adlandırılır), dolayısıyla *ortak anahtar*

algoritması olarak da adlandırılan algoritma sınıfının adıdır. Kamu terimi hakkında daha fazla bilgi daha sonra tartışılacaktır.

Şifrelemek için kullanılan işlev, şifresini çözmek için kullanılanla aynı olmak zorunda değildir. Önemli olan, birlikte şifreleme algoritmasını temsil etmeleridir, bir kısmı bir anahtara dayanır ve diğer kısmı diğer anahtara dayanır.

İki anahtar o kadar matematiksel özelliklere sahiptir ki, bir anahtarı diğerinden keşfetmek oldukça karmaşıktır ve muazzam miktarda zaman gerektirir. Özellikle, ortak anahtarı, *Düz Metin* ve *CypherText'i* bilseniz bile, özel anahtarı yeniden oluşturmak makul olmayan bir süre gerektirir.

## Asimetrik Algoritma Kullanımları

Ortak *anahtarın* herkese açık olduğunu ve sahibinin karşılık gelen özel anahtarı bilen tek kişi olduğunu varsayalım (ayrıntılar daha sonra verilecektir). Herkes tarafından kullanılabilen ortak anahtar olarak, herkes bir mesajı şifrelemek için kullanabilir, ancak yalnızca ilgili özel anahtarın sahibi mesajın şifresini çözebilir, bu da mesaj iletiminin gizliliğini sağlar.

Bu nedenle, bir öznenin diğer *birçok* özneyle gizli bir şekilde iletişim kurmak için n farklı anahtara ihtiyaç duyduğu bir durumdan, herkesin bir konuya gizli bir mesaj göndermesi için yalnızca bir anahtarın (alıcının ortak anahtarı) gerekli olduğu bir duruma geçtik.

*N* deneğin birbiriyle iletişim kurmasına izin vermek için, her öznenin bir çift anahtarı olmalıdır, bu nedenle toplam anahtar sayısı 2·n'dir. İlginç olan nokta, anahtarların sayısının azalması değil; ancak açık anahtarlar olanse, çok daha basit bir şekilde iletilebilir. Başka biri bu açık anahtarları tanırsa, bu hiç sorun değildir. Bu nedenle, bu anahtarlar gerçekten yayınlanabilir, herkesin kullanımına sunulabilir ve herkes mesajların korumasını zayıflatmadan bunları kullanabilir.

Özne, genel karşılığı herkese açık olan özel (gizli) bir anahtarın tek sahibiyse, herkes konunun özel anahtarıyla şifrelenmiş iletileri doğrulayabilir (şifresini çözebilir). Daha sonra, bir mesajın şifresini çözmek için bir konunun ortak anahtarını kullanmak mümkünse, yalnızca karşılık gelen özel anahtara sahip olan öznenin mesajı şifreleyebileceği anlamına gelir, bu da mesajın **reddedilmemesini** sağlar.

Bir sorun hala çözülmesi gerekiyor: bir ortak anahtarın meşru konuyla gerçekten ilişkili olmasını sağlamak. Burada bir *süper partes* sertifikasyon yöntemi sağlanmalıdır, bu daha sonra *tartışılacak bir Ortak Anahtar Altyapısı* (PKI) tarafından sağlanır.

Bir yan not olarak, asimetrik algoritmalar bir *meydan okuma-yanıt algoritması* olarak da kullanılabilir. A'nın öznenin *doğrulanması* (bir öznenin gerçekten B olduğunu iddia eden kişi olduğunu kanıtlaması) için, A (açıkça) B'ye rastgele bir sayı gönderir; sonra B, onu özel anahtarıyla şifreler ve A'ya geri gönderir. A, mesajın şifresini B'nin ortak anahtarıyla çözebiliyorsa, A, diğer tarafın B

olduğundan emindir, çünkü B'nin ortak anahtarının şifresini çözebileceği bir şeyi şifreleyebilen tek kişi budur.

**Avantajları ve Dezavantajları**

Toplam anahtar sayısı önemli ölçüde azalır, bu da tüm anahtarların doğru yönetimini garanti etmenin karmaşıklığını azaltır (ve yalnızca sahibi gizli anahtarı bilir). Ortak anahtarlar herkese açık hale getirildiğinden, farklı konular arasında belirli anlaşmalara gerek yoktur. Örneğin, açık anahtarların bu şekilde kullanılması, e-ticarette çok önemlidir, çünkü e-ticaret, satıcının müşterilerle a priori bir anlaşması olmasını gerektiremez.

İki anahtar arasındaki bu bağlantıyı elde etmek ve aynı zamanda birinin diğerinden hesaplanamayacağını ve biriyle şifrelenen şeyin diğeriyle şifresinin çözülmesini garanti etmek, asimetrik algoritmaları matematiksel problemler açısından simetrik algoritmalardan çok daha karmaşık hale getirir. Bu matematiksel problemler temel olarak simetrik algoritmalardan daha yavaş algoritmalar ve en azından şu anda kullanılan algoritmalarla daha uzun anahtarların kullanılmasını gerektirir. Bu nedenle, bir yandan daha fazla esneklik varsa, diğer yandan bilgi işlem kaynakları açısından daha zahmetlidirler.

***Öne çıkan algoritma: RSA*** *(Ronald R ivest, Adi* ***S hamir****ve Leonard* ***A dleman'ın*** baş harflerinden*)*
Bu algoritma, asal faktörizasyonun karmaşıklığına dayanır: iki asalın çarpılması ve sonucu elde edilmesi kolaydır, ancak sonuçtan iki asal bulmak karmaşıktır / inatçıdır. Asal faktörizasyon zor bir problemdir. Ayrıca, çok yüksek asal sayıları bulmanın kolay olmadığını da biliyoruz, çünkü asal sayıları bulmaya, onları hesaba katmaya çalışmaktan ve asal olup olmadıklarını keşfetmekten başka bir kural yoktur. Bu, çok büyük sayılar üzerinde çalışmanın, belirli bir sayının türetildiği iki asal sayının hangisi olduğunu bulmanın, önemli bilgi işlem kaynakları gerektiren bir sorun olduğu anlamına gelir. Öte yandan, meşru işlemler sınırlı bilgi işlem kaynaklarıyla yapılabilir.

***Öne çıkan algoritma:*** Eliptik Kriptografi *(ECC - Eliptik Eğri Kriptografisi)*

Bu durumda "zor problem", ayrık logaritmanın (sonlu bir alanda) hesaplanmasıdır. "Sonlu alan", tamsayı değerlerinin sınırlı bir kümede olduğu anlamına gelir (çok kaba bir tanım).

ECC problemi asal faktörizasyondan daha karmaşıktır, bu nedenle daha kısa anahtarlarla aynı güvenliği elde edersiniz. ECC algoritmaları RSA'dan daha hızlıdır.

**Simetrik ve Asimetrik Algoritmaların Karşılaştırılması**

Asimetrik algoritmalar, simetrik kriptografi dışındaki matematiksel problemlerden yararlanır. Bu, anahtarların uzunluklarının simetrik kriptografininkilerle karşılaştırılamayacağı anlamına gelir. Simetrik şifreleme ile ilgili sorun, anahtarın bir anahtarla eşleşebilecek tüm olası sayıların

denenmesini önleyecek kadar uzun olmasıdır, bu nedenle 128 bitlik bir anahtarla $2^{128}$ (yani $3.4 \cdot 10^{38}$) olası anahtar vardır. Günümüzün simetrik algoritmaları tipik olarak 128, 256 ve bazen 512 bit anahtarlar kullanır.

Örneğin, asal sayılardan yararlanan RSA'yı düşünürsek, tüm anahtarların mümkün olmadığı açıktır, çünkü bunlar asal sayıların varlığıyla ilgilidir. Tüm sayılar asal değildir, bu nedenle tüm sayılar olası anahtarlar değildir. Çok basitleştirici, 128 bitlik bir simetrik algoritmanın aynı algoritma sağlamlığına (dolayısıyla hesaplama açısından aynı zorluk) sahip olması için, RSA algoritmasının 1024 bit sırasına sahip bir anahtara ihtiyacı vardır. Günümüzün asimetrik algoritmaları tipik olarak 1024, 2048 ve 4096 bit anahtarları kullanır.

## Hash İşlevleri

Karma işlevleri, rasgele uzunluktaki bir metnin kısa bir sabit uzunlukta özetini hesaplar; bu özete *özet* denir. Bir hash fonksiyonunun hesaplanması basit ve hızlı olmalıdır.

### Güvenlik özellikleri

Hash fonksiyonları, onu oluşturan *olası* bir metni kurtarmak için özetten çok karmaşık ve yavaş olması anlamında "tek yönlü fonksiyonlar" dır, ancak bu metin pek kullanışlı değildir. Bu, orijinal metin özetinin özetten daha büyük olması (ve olması gerektiği) gerçeğinin bir sonucudur, bu nedenle neredeyse sonsuz bayt dizileri belirli bir özetle sonuçlanabilir. Bir metnin verilen özete sahip olduğu tespit edilse bile, orijinal metin olması veya hatta mantıklı olması son derece imkansızdır. Aynı özete sahip iki anlamlı mesaj oluşturmak veyabelirli bir özetten anlamlı bir mesaj çıkarmak son derece zordur, bir mesajın herhangi bir modifikasyonu, hatta tek bir bit bile, tamamen farklı bir özet verecektir (bitlerinin yaklaşık% 50'si farklıdır).

### Bütünlük kontrolü

Bir M mesajı verildiğinde, D özeti bir hash fonksiyonu H: D = H(M) aracılığıyla hesaplanır.

D özetinin güvenli bir şekilde iletilmesi, M mesajının doğru olmasını sağlar: M mesajının alıcısı, üzerindeki özeti hesaplar ve alınan (D) ile karşılaştırır; aynı ise, ileti değiştirilmemiştir. Sindirimin güvenli bir şekilde iletilmesi esastır, aksi takdirde hiçbir koruma yoktur.

Bu mekanizma, ağ protokollerinde yaygın olarak kullanılan CRC'ler (Döngüsel Artıklık Denetimleri) olarak bilinen diğer daha az güçlü (ancak daha hızlı) algoritmalar yerine indirilen bir yazılımın doğrulanması olarak da kullanılır.

### Bazı Önemli Karma Algoritmalar

Aşağıdakiler, en bilinen karma algoritmalardan ikisidir.

*MD5 (İleti Özeti #5, 1991)*

MD5, düşük bir hesaplama gücü gerektirir (diğer karma algoritmaların çoğuna kıyasla) ve 128 bitlik bir karma üretir. Zamanla, güvenlik amaçları için tamamen güvenilmez olacak şekilde yavaş yavaş zayıfladı (şu anda saniyeler içinde çarpışma olarak adlandırılan aynı MD5 karmasıyla iki ayrı mesaj bulmak mümkündür ), yine de kriptografik olmayan uygulamalarda, örneğin daha yaygın (ve zayıf, ama daha hızlı) CRC-32.

*SHA-3 (Güvenli Karma Algoritma #3, 2015)*

SHA aslında güvenli karma algoritmaların bir ailesidir, SHA-3 son sürümdür ve kriptograflar arasındaki rekabetin sonucudur. Algoritma, güvenliği hız ile dengeleyerek ayarlanabilir; 224, 256, 334 ve 512 bitlik hash değerleri üretir. MD5'ten 2-3 kat daha yavaştır, ancak şu ana kadar herhangi bir zayıflık bulunmamıştır.

## İmzalar (Dijital İmzalar)

Bazı ülkelerin mevzuatında elektronik imza olarak adlandırılan dijital imzalar, bir mesajı alır, bir karma hesaplar ve asimetrik bir algoritmanın gizli anahtarıyla (sadece karma) şifreler. Gizli anahtarla imzalanan bu karmaya *imza* adı verilir. Dijital imza daha sonra orijinal mesajı şifrelemez, yalnızca ilgili metnin değiştirilmediğini garanti etmeye izin verir, çünkü aksi takdirde özet değişir. Ayrıca, aslında asimetrik bir algoritmanın özel anahtarıyla şifrelendiği için kaynağın doğrulanabileceğini (reddedilmemesi) garanti eder. Bu, herkesin metnin sağlam olduğunu ve aslında belirli bir konu tarafından oluşturulduğunu doğrulayabileceği anlamına gelir.

Dijital imzalara benzer şekilde **İleti Kimlik Doğrulama Kodları** (MAC'ler) bulunur. Ayrıca bir mesajın bütünlüğünü ve özgünlüğünü sağlamayı da amaçlarlar. Özet, korunacak metnin ve bir *simetrik algoritma* anahtarının (reddedilmeme güvencesi yoktur, bu nedenle yalnızca bu özelliğe ihtiyaç duyulmadığında uygundur) karma olarak hesaplanır. En bilinen algoritmalardan biri HMAC: Anahtarlı karma MAC'tir.

## Ortak Anahtar Altyapıları

Asimetrik şifreleme ve dijital imza işlemlerinin gizlilik, bütünlük ve özgünlük gereksinimleri, ilgili taraflar arasında bir güven ilişkisi kurulmasını gerektirir. Tarafların daha önce hiçbir teması olmadığı gibi, güvenilir (onlar tarafından) bir üçüncü taraf gerekli garantileri sağlamalıdır, başka bir deyişle bu üçüncü taraf bir ortak anahtarın sahibinin orijinalliğini sağlamalıdır.

*Sertifika Yetkilisi* (CA), kullanıcının kimliğini garanti eden hiyerarşik güven modelinin bir parçasıdır. İletişimde yer alan tüm kuruluşlar tarafından "güvenilir" olarak tanınmalıdır.

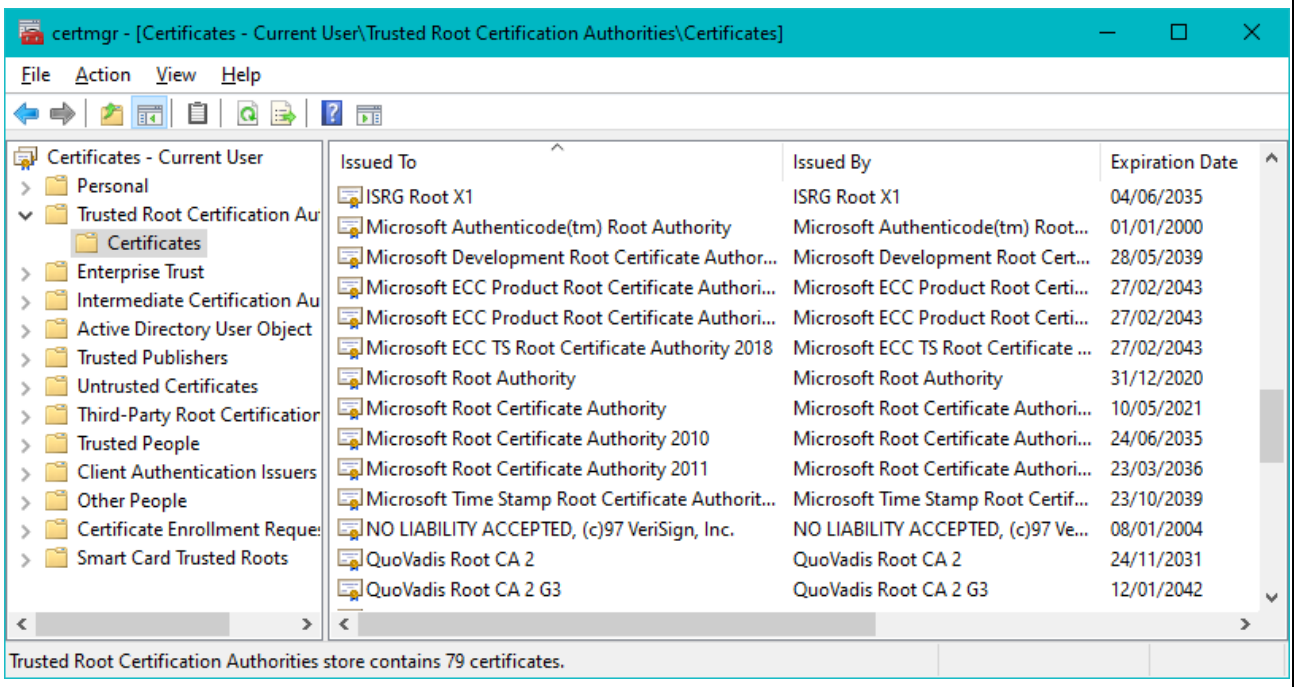
Bir öznenin gerçekliğinden emin olmak için, CA, özneyi ortak anahtarına bağlayan dijital bir sertifika imzalar (kavramsal olarak bir kişinin Kimlik belgesine benzer). CA tarafından uygulanan imza dijital

bir imzadır, bu nedenle CA'nın özel anahtarı kullanılarak hesaplanır. Elbette, dijital sertifikanın CA'nın ortak anahtarı kullanılarak doğrulanması gerekir.

*Ortak Anahtar Altyapısı* (PKI), her CA'nın ortak anahtarının önceki düzey CA (bu amaçla özel anahtarını kullanan) tarafından imzalandığı hiyerarşik bir CA'lar yapısıdır; işlem, özel anahtarını kullanarak ortak anahtarını tutan bir sertifikayı kendi kendine imzalayan Kök Sertifika Yetkilisi'ne kadar uzanır.

Hiyerarşik bir model, herhangi bir derinlikte hiyerarşiler oluşturmaya izin verir. Örneğin, bir Ülke içinde, ulusal bir kök CA'ya kadar belediye, ..., bölgesel sertifikasyon yetkililerine sahip olabiliriz.

Kök CA sertifikasının, sahte bir sertifikayla değiştirilmesini imkansız hale getirmek için birçok yerde ve şekilde genel kullanıma sunulması gerekir. Örnek olarak, İşletim Sistemlerinin kendileri bir dizi kök ve güvenilir ara CA ile birlikte gönderilir. Şekil 1, MS Windows 10 işletim sisteminden Kök Sertifikalar listesinin bir alıntısını göstermektedir.




Şekil 1. Windows 10 Sertifika Yöneticisi

## PKI'nın bileşenleri

Ana aktör **Sertifika Yetkilisi'dir (CA),** PKI'nın şifreleme anahtarlarını ve sertifikalarını yönetir. Gerçek anahtarları tutan ve daha sonra şifreleme işlemlerini gerçekleştiren bileşendir. Açıkçası, bu anlamda aynı zamanda sertifika yetkilisinin en kritik unsurudur.

**Kayıt Yetkilisi (RA)** sertifika isteklerini yönetir ve verilen sertifikaları sağlar. Kriptografik belirteçlerin ve yazılımların dağıtımı ile ilgilenir (CA'nın ön ucudur). Örneğin, bir vatandaşın dijital

bir sertifika talep etmek için gidebileceği ve elde edilebileceği bir sayaç gibidir. Fiziksel bir sayaç veya bir web sitesi olabilir, tür farklı güvenlik düzeyleriyle ilişkilidir. Buna ek olarak, bir RA genellikle kullanıcıya anahtar çiftini oluşturmada ve bu sertifikaları güvenli bir şekilde yönetmede yardımcı olabilir, örneğin başvuru sahibini imza oluşturma, belgeleri şifreleme, imzaları doğrulama ve bilgilerin şifresini çözme konusunda destekleyen geçici yazılım veya donanım araçları sağlayarak.

**Sertifika Sunucusu**, kullanıcı sertifikalarının yayımlandığı ve bir şekilde aranabilir hale getirildiği dizindir. Dizin X.500 standardını izler ve LDAP protokolü kullanılarak sorgulanır.

## Dijital Sertifika

Dijital sertifika yapısı ve formatı ITU-T X.509 standardı ile tanımlanmıştır (X.509 numaralı Uluslararası Telekomünikasyon Birliği'nin bir standardıdır).

X.509 dijital sertifikası en az aşağıdaki alanları içermelidir:

**DN** (ayırt edici isim) kullanıcı: Gaius Julius Caesar

**CN** (ortak adı): Jül Sezar

**O** (organizasyon): Hükümet

**OU** (organizasyon birliği): Senato

**C** (ülke): Roma İmparatorluğu

Bu, X.509'un ailenin bir üyesi olduğu X.500 standardını karşılayan bir sertifikanın klasik desenidir. Bir sertifikanın başka öznitelikleri de olabilir (ör. e-posta). X.500 Dizin yoluna karşılık gelen bir IP adresi veya URL olabilir (ör. LDAP – Basit Dizin Erişim Protokolü):

Bilişim / Roma İmparatorluğu / Hükümet / Jül Sezar / Gaius Jül Sezar /

X.509 dijital sertifikası en azından aşağıdaki alanları da içermelidir:

**DN Veren**: Yetkilinin X.500 biçimindeki adı (sertifika yetkilisinin sertifikada görünen kendi ayırt edici adı da vardır. Açıkçası, temsil edildiği biçim, sertifikanın geçerli olduğu konunun ayırt edici adıyla aynıdır.

**Seri Numarası**: Kurum tarafından tutulan sertifikanın seri numarası (her sertifika, yetkili tarafından kendisine atanan seri numarası ile benzersiz şekilde tanımlanır. Öte yandan, çift (*seri numarası* ve *verenin ayırt edici adı*), dünya çapında benzersiz bir çift olarak verilebilecek sertifikaların bolluğu boyunca bir sertifikayı benzersiz bir şekilde tanımlar).

**NotBefore**: etkinleştirme tarihi

**NotAfter**: son kullanma tarihi

Bunlar kesinlikle birbiriyle ilişkili iki bilgidir. Bu nedenle, bir sertifikanın yaşamın başlangıcı ve artık geçerli olmadığı bir son kullanma tarihi vardır. Bunun nedeni, anahtarları aşırı kullanımdan koruma arzusundan kaynaklanmaktadır.

**Ortak Anahtar**: sertifikalı öznenin ortak anahtarı (sertifika kişisel verileri ve ortak anahtarı içerir)

**Ortak Anahtar** Algoritması: Ortak anahtarın uygun olduğu algoritma

**İmza** Algoritması: Yetkili tarafından Sertifikayı imzalamak için kullanılan algoritma

**İmza**: Sertifika Yetkilisi tarafından Sertifikaya yapıştırılan imza (bu nedenle sertifikanın kendisinin orijinalliğini garanti eden öğe)

**KeyUsage:** anahtarın kullanılabileceği amaçlar

## Sertifikaların Oluşturulması ve İptali

RFC 2314 standardı (PKCS # 10'dan) *certificateRequest* iletisinin biçimini tanımlar ve başvuru sahibinin ortak anahtarını ve verilerini içerir (her ikisi de bir RA'ya verilir). İstek mesajı, özel *anahtara sahip olunduğunun sözde kanıtını* gerektirir : certificateRequest mesajı, başvuru sahibinin özel anahtarıyla imzalanmalıdır. Yetkili makam, bu imzayı doğrulayarak, başvuru sahibinin bu anahtara gerçekten sahip olduğunu büyük ölçüde doğrulamaktadır (aksi takdirde başvuran bu imzayı üretemezdi).

"NotAfter" özniteliğine karşılık gelen tarihte, Sertifika "süresi dolmuş" olarak kabul edilir. Süre sonu, dijital sertifikayla ilişkili aynı asimetrik anahtarların aşırı bir süre boyunca kullanılamamasını sağlamaya da yarayan ve bu da onları bir dizi kriptoanalitik saldırıya maruz bırakabilecek bir mekanizmadır.

Süre sonu şu anlamlara gelebilir:

- **sertifika** (ilişkilendirme konusu-anahtarı)
- **ilişkili anahtarlar**, amaçlandıkları kullanımla ilgili olarak (bir anahtar yalnızca belirli hedefler için kullanılabilir, diğerleri için kullanılamaz)

Süresi dolmuş bir sertifika, sahibi tarafından kullanılamaz, çünkü süresi dolduğu için kullanımı pratikte yasaktır. Gerçekte, hiçbir şey sertifika sahibinin başka amaçlar için kullanmasını engellemez, ancak onunla iletişim kuran herkes bir şekilde bu sertifikanın süresinin dolduğunu tespit edecek ve kabul etmeyecektir. Sertifika elbette süresi dolmadan önce verilen iletileri doğrulamak ve şifrelerini çözmek için kullanılabilir. Bazı durumlarda, genellikle otomatik olmayan bir sertifikayı "yenilemek" mümkündür .

Bazen bir sertifika, *iptal* yoluyla sona ermeden önce geçersiz kılınabilir (bu, "notAfter" öznitelik değerinden önce gerçekleşir. Bunun nedeni, örneğin, sertifikayla ilişkili anahtarın güvenliğinin ihlal

edilmiş olması, öznenin artık şirketle veya değiştirilen birimle ilişkili olmaması, sertifikayı veren CA'nın (veya daha üst düzey CA'larından birinin) güvenliğinin ihlal edilmiş olması veya başka nedenlerle olabilir. İptal edilen bir sertifika, yalnızca "beklemede" tutulduğu durumlar dışında artık kullanılamaz. T, sertifikanın belirli nedenlerden dolayı ("olası anahtar uzlaşması", "mal sahibi dışında bir kişi tarafından iptal talebi", "sertifikanın geçici olarak kullanılmaması" gibi nedenlerle CRL'ye geçici olarak eklendiği belirli bir iptal şeklidir. Askıya alma işleminden "yeniden etkinleştirme" durumunda, CRL'ye yeni bir giriş, özel bir neden koduyla eklenir: *CRL'den* hiçbir giriş silinemeyeceğinden, removeFromCRL.

PKI kullanıcılarına sertifikanın geçerliliğini kaybettiğini bildirmek için, seri numarası *Sertifika İptal Listesi'nde* (CRL) listelenir. Bu liste, sertifikayı veren Sertifika Yetkilisi tarafından düzenli aralıklarla yayımlanır ve tüm kullanıcılara (aktif veya pasif olarak) yayılır, böylece böyle bir sertifikaya sahip olan tüm özneler artık onu kullanamaz. CRL, ITU-T X.509 standardına uygundur - rfc3280 - rfc4325, bu da formatının standartlaştırıldığı anlamına gelir. CA tarafından düzenli aralıklarla imzalanır ve yeni Sertifika İptal Listesi'nin kullanılabileceği tarih ve saati içeren "nextUpdate" adlı bir alan içerir. Bu tarih geçmişe aitse, CRL'mizin "süresi dolmuş" demektir, tarih geleceğe aitse, iptal edilen sertifikalar listemizi ne zaman güncellememiz gerektiğini biliyoruz. İptal edilen her sertifika, iptal nedenini içerir. CRL, LDAP protokolü aracılığıyla bir hizmet olarak sağlanır.

CRL'ler oldukça büyüyebildiğinden, özellikle sertifika yetkilisinin yüz binlerce, hatta milyonlarca kullanıcısı varsa, bölümlenebilir ve CA, örneğin belirli bir süreye atıfta bulunarak Sertifika İptal Listesi'nin yalnızca parçalarını sağlar. Bu nedenle, bir sertifikayı doğrulamak gerektiğinde, vatandaş ilgilendiği tek CRL parçasını indirir. CRL'yi veya bir bölümünü karşıdan yüklemek yerine, CRL'de belirli bir sertifika numarasının olup olmadığını bulmak için *Çevrimiçi Sertifika Durumu Protokolü'nü* (OCSP - RFC 2560) uygulayan bir hizmet kullanılabilir. Yanıtlar, güvenilmek üzere CA tarafından imzalanır

## Güven Zinciri

Bir sertifikayı doğrulamak için, öznenin güvenilir bir CA'dan, muhtemelen kök CA'dan başlayarak ara CA'ların tüm zincirini doğrulaması gerekir. Önce denetlenecek sertifika sertifikayı veren CA'ya karşı doğrulanır, ardından CA sertifikası üst CA'sına karşı doğrulanır ve böylece bir ara güvenilen (kullanıcı tarafından) CA'ya veya kök CA'nın kendisine kadar devam eder.

## Anahtar Kullanımı

Aslında, anahtarlar farklı amaçlar için kullanılabilir ve bir sertifika izin verilen kullanımları belirtir, bunların arasında şunlar vardır:

- **Reddetmeme: Anahtar, yasal değeri olan** bir imza üretmek için kullanılabilir ve daha sonra üçüncü taraflara el yazısı imza olarak karşı çıkılabilir.
- **dataEncipherment:** anahtar verileri şifrelemek için kullanılabilir
- **keyAgreement:** anahtar, anahtarları güvenli bir şekilde değiştirmek için kullanılabilir

## Zaman Damgası

PKI, geçmişteki bir noktada varlığını kanıtlamak için bir belgeyi Zaman Damgası Protokolü'ne (RFC 3161) göre zaman damgalamak için de kullanılabilir. CA belgeyi imzalar ve zamansal bir işaret içerir. İmza, zaman damgası yapıştırıldıktan sonra belgenin değiştirilmemesini de sağlar. Bir ZamanDamgası, elektronik imzanın geçerliliğini, kullanılan anahtarlarla ilişkili dijital sertifikanın ömrünün ötesinde, örneğin geçen her yıl sözleşmeye yeni bir zaman damgasının eklenmesini talep ederek, extend edebilir. Yapıştırma, her yıl imzanın uygulandığı süre için yeterli güçte kriptografik teknikler kullanılarak yapılacaktır.

## Standart Mekanizmalar ve Formatlar

Aşağıdakiler, yaygın olarak kullanılan iki standart mekanizma ve formattır.

*PKCS #7 - Ortak anahtar şifrelemesi standardı #7*

PKCS #7, imzalanmış, şifrelenmiş (simetrik ve asimetrik şifreleme ile) ve MAC tarafından doğrulanabilir (sindirilmiş veriler) mesajların / dosyaların biçimini tanımlar.

*S/MIME - Güvenli Çok Amaçlı Posta Uzantısı*

Güvenli Çok Amaçlı Posta Uzantısı (RFC 3851 - SMIMEv3), MIME standardının (RFC 822'nin bir uzantısı olarak RFC 2045, 2046 ve 2047) bir uzantısıdır ve posta iletilerinin şifrelenmesini açıklar. PKCS #7 varlıkları büyük ölçüde yeni MIME varlıklarına çevrilir ve bu nedenle imzalı veya şifreli e-posta iletilerinin veya imzalı veya şifreli eklerin bölümlerini bulabileceğimiz yeni kapsayıcıları tanımlar.

## Exchange ve Depolama Biçimleri

PKCS #12 ( **Kişisel Bilgi Değişimi Sözdizimi Standardı #12),** şifreleme anahtarlarının ve dijital sertifikaların (güvenilir sertifika zinciri) değişimi ve güvenli depolanması (dosyada) için bir biçim tanımlar. Depolama güvenlidir, çünkü hem anahtarların hem de sertifikaların ve bir öznenin güvenmeyi seçtiği tüm sertifika zincirinin şifrelenmesine dayanır. Bir kullanıcı bir imza ve şifreleme anahtarı ile sertifikayı dışa aktardığında, örneğin yedek kopya veya başka bir aygıta kopyalamak için tarayıcılar, e-posta istemcileri ve işletim sistemleri tarafından kullanılan biçimdir.

PKCS #12, verileri anahtarı kullanıcı tarafından sağlanan bir paroladan türetilen simetrik bir algoritmayla şifreler. İçeriğin bütünlüğünü HMAC veya dijital imza aracılığıyla garanti eder.

PKCS #12 kullanmanın faydaları şunlardır:

- birçok cihaz için birçok kütüphanede bulunan bir format olduğu için uygulama maliyeti yoktur (bilgisayarlardan cep telefonlarına)
- **çoğu işletim sistemi ve yazılım tarafından zaten destekleniyorlar**, bu da hemen kullanılabilir olduğu anlamına geliyor.

- **"güçlü" simetrik algoritmalar kullanır** (uygulanan şifreleme güçlü, çok dayanıklı bir şifrelemedir, bu nedenle örneğin kaba kuvvet saldırılarına direnen anahtarlardır).
- **farklı cihazlarda taşınabilir**, bir dosyayı diğer cihazlara taşımak veya anahtarların her zaman kullanılabilir olması için kullandığımız tüm cihazlara kopyalamak mümkündür.

Bazı dezavantajları da vardır:

- **silinebilir, kopyalanabilir, iletilebilir,** bu önemli bir sorundur çünkü çalınabilir, silinebilir, uzaklaştırmak için kopyalanabilir veya kullanılamaz hale getirilebilir.
- **özel anahtar doğrudan imzalama / şifre çözme yazılımı tarafından kullanılmalıdır, bu,** yazılımın kullandığı yazılımın bu dosyanın şifresini çözdüğü ve anahtara doğrudan eriştiği anlamına gelir, bu nedenle anahtarın sistemde açık, korumasız olduğu bir an vardır, belki de en az erişilebilir alan olan geçici bellektir; ancak sistemde bir şekilde açık metin olarak mevcuttur ve bu çok önemlidir. Bu formatın güçlü sınırlaması
- **kaba kuvvet saldırıları gerçekleştirmek mümkündür,** örneğin, birisinin PKCS #12 dosyasını çalması durumunda, olası tüm şifreleri ayrı bir bilgisayar sisteminde deneyebilirler.

## Kriptografik Jetonlar

Kriptografik tokens , şifreleme algoritmalarını yerel olarak uygulayan donanım aygıtlarıdır. Kriptografik malzemeye dışarıdan doğrudan erişime izin vermezler, anahtara erişmek ve anahtarı kullanmak için yalnızca etkinleştirilirler. Ayrıca, bu komutlar genellikle yalnızca kullanıcı ve kullanıcı adına sistem, örneğin bir PIN girerek belirteçte kimlik doğrulaması yaptıktan sonra yürütülebilir. Belirteçler, kullanıcının anahtar oluşturma, şifreleme ve veri imza işlemlerini istediği küçük bir İşletim Sistemine sahip olabilir. Belirtecin güvenli bir şekilde imha edilmesi genellikle sadece iptal değil, aynı zamanda anahtarın yazıldığı hafıza alanlarına erişememe anlamına gelir, bu nedenle bugün düşünemediğimiz herhangi bir saldırı biçiminden kaçınmak için. İşletim sistemi ve belirteç donanımı belirli güvenlik standartlarına göre sertifikalandırılabilir.

## Meydan Okuma-Yanıt Mekanizması

Bir meydan okuma/yanıt mekanizması, iki tarafın iletişim kanalında yararlı bilgiler açığa çıkarmadan kimlik doğrulaması yapmasına olanak tanır. Buradaki fikir, bir taraf A'nın diğer taraf B'ye , örneğin Tek Yönlü bir H fonksiyonu ile kendi anahtarıyla, rastgele bir sayı (sadece bir kez kullanılan bir sayı olduğu için nonce olarak adlandırılır) üzerinde çalışmasını "zorlar". Sonra B, H(nonce)'u A'ya döndürür. B, A'nın beklediğini döndürürse, A, B'nin kimliğinden emindir. En basit durumda, B, nonce'yi A tarafından bilinen gizli bir anahtarla şifreler. Aynı anahtara sahip olan A, aynı anahtarı kullanarak fonksiyonu nonce'ye uygular ve B tarafından kendisine verilen sayının hesapladığı sayıyla eşleştiğini doğrular.

Bu, kanal üzerinden gönderilmemiş bir parolayı temel alan bir kimlik doğrulamasıdır, bu nedenle anahtarı izlemek mümkün değildir. Mekanizmayı etkili kılan şey, rastgele sayının her seans için farklı olmasıdır. Kanalda okunan bilgiler her oturum için farklı olduğundan, sonraki kimlik doğrulaması için kullanışlı değildir. Bir avantajı, senkronizasyona gerek olmamasıdır (kullanılacak Pad'in başlangıç noktasını bilmenin önemli olduğu Tek Kullanımlık Parola mekanizmasından farklı olarak). İnsanlar tarafından kullanılması kolay olmadığından, genellikle makineden makineye kimlik doğrulamasında kullanılır. ADSL yönlendiricilerinin sağlayıcı kimlik doğrulama sunucusuna ağ kimlik doğrulama mekanizmalarında bir sınama-yanıt mekanizması bulunur.

## Soru:

Ayrı bir dosyada.

## Ekler:

Powerpoint ayrı bir dosyada.

## Başvuru:

Her konunun Vikipedi'de kendi sayfası vardır, bazen bu dersin amaçları için zaten çok kapsamlıdır; Kitaplar çok uzmanlaşmıştır ve ancak belirli bir konuda çok derin bir eğitimden sonra yararlıdır.