

# ESYA API SmartCard

## Kullanım Kılavuzu

versiyon 0.2

*Kriptografik işlemlerin güvenli bir ortamda yapılması amacıyla akıllı kartlara ihtiyaç duyulmaktadır. Akıllı kartlar özel anahtarın (private key) dışarıdan erişilmesine izin vermeyerek açık anahtar altyapısı için gerekli güvenliğini sağlarlar.*

*Akıllı kart içinde kullanıcının sertifikaları, özel anahtarları ve açık anahtarları bulunmaktadır. Her sertifikanın bir açık anahtarı ve bir özel anahtarı yine kart içinde yer almaktadır. Sertifikalar ve açık anahtarlar kart içinden okunabilmektedir. Özel anahtar ise dışarı kart dışına çıkartılamaz, anahtar ile kart içinde kriptografik işlemler yapılabilir.*

*ESYA API SmartCard modülü akıllı kart ile işlemlerine yardımcı olur, PKCS7 yapısında basit imza atabilir.*

# çindekiler

<b>Bölüm 1 Giri</b>	<b>1</b>
<b>Bölüm 2 Gereksinimler</b>	<b>1</b>
<b>Bölüm 3 Akıllı Karta Eri im</b>	<b>1</b>
1 Akis Kartlara Eri im .....	2
<b>Bölüm 4 Akıllı Kartın Sertifikasının Okunması</b>	<b>3</b>
<b>Bölüm 5 Akıllı Karttaki Nesne Adlarının Okunması</b>	<b>4</b>
<b>Bölüm 6 Akıllı Karta mza atılma- ifreleme lemlerinin Yapılması</b>	<b>5</b>
<b>Bölüm 7 PKCS7 Yapısında mza Atılması</b>	<b>6</b>
<b>Bölüm 8 Lisans Ayarları</b>	<b>7</b>
1 Test Lisansı ile Çalışma .....	7

# 1 Giri

Kriptografik i lemlerin güvenli bir ortamda yapılması amacıyla akıllı kartlara ihtiyaç duyulmaktadır. Akıllı kartlar özel anahtarın (private key) dı arıdan eri lmesine izin vermeyerek açık anahtar altyapısı için gerekli güvenli i sa larlar.

Akıllı kart içinde kullanıcının sertifikaları, özel anahtarları ve açık anahtarları bulunmaktadır. Her sertifikanın bir açık anahtarı ve bir özel anahtarı yine kart içinde yer almaktadır. Sertifikalar ve açık anahtarlar kart içinden okunabilmektedir. Özel anahtar ise dı arı kart dı na çıkartılamaz, anahtar ile kart içinde kriptografik i lemler yapılabilir.

ESYA API SmartCard modülü akıllı kart i lemlerine yardımcı olur, PKCS7 yapısında basit imza atabilir.

# 2 Gereksinimler

SmartCard API'si "ma3api-smartcard-....jar" ve "ma3api-common-....jar" kütüphanelerine ihtiyaç duymaktadır. Ayrıca kullanılacak akıllı kartın ve akıllı kart okuyucusunun sürücüsü sisteme kurulmu olması gerekmektedir.

.NET SmartCard API'si ise "ma3api-smartcard.dll" kütüphanesinin yanısıra ba ımlı oldu u "ma3api-asn.dll", "ma3api-common.dll", " ma3api-crypto.dll", "asn1rt.dll", ",ma3api-crypto-bouncyprowider.dll", "ma3api-iaik\_wrapper.dll", "log4net.dll" ve "ma3api-pkcs11net.dll" kütüphanelerine ihtiyaç duymaktadır.

# 3 Akıllı Karta Eri im

*SmartCard* sınıfı akıllı kart ile ilgili i lemlerden sorumlu sınıftır. *SmartCard* sınıfının çalış ırılabilmesi için hangi akıllı kartın kullanıldı ının bilinmesi gerekmektedir. Çünkü akıllı kart i lemleri akıllı kartın sürücüsü üzerinden yapılmaktadır. Java 6 ile java kütüphaneleri kullanılarak kart bilgilerine eri ilip hangi kart oldu u belirlenebilmektedir. Java 5 ve .NET'te ise hangi akıllı kartın kullanıldı ı bilinmelidir.

Java 6 için *SmartOp* sınıfının *findCardTypeAndSlot()* fonksiyonu ile kartın slot numarası ve kart tipi bulunabilmektedir. E er bilgisayara bir akıllı kart takılı ise fonksiyon do rudan bu kartın bilgilerini dönecektir. E er birden fazla akıllı kart takılı ise *javax.swing . JOptionPane* ile kullanıcıya akıllı kartlardan biri seçtirilecektir.

Java6:

```
Pair<Long, CardType> slotAndCardType = SmartOp.findCardTypeAndSlot();
Long slot = slotAndCardType.getObject1();
SmartCard smartCard = new SmartCard(slotAndCardType.getObject2());
long session = smartCard.openSession(slot);
```

E er görsel bir arayüzün API tarafından gösterilmesini istemiyorsanız; *SmartOp* sınıfının *getCardTerminals()* fonksiyonu ile akıllı kart okuyucularının isimlerini alabilirsiniz. Bu

isimler ile kartı kullanıcıya seçtirdikten sonra *getSlotAndCardType(String terminal)* fonksiyonuyla kullanıcının seçtiği kartın slot numarası ve kart tipi alınabilir.

Eğer kullanıcıya kart tipine göre akıllı kartı seçtirmek isteniyorsa, *SmartOp* sınıfının *findCardTypesAndSlots()* ile başlı olan bütün kartların slot numaralarını ve kart tiplerini alabilirsiniz.

Java 5 ve .NET'te ise akıllı kartın türü bilinmelidir. Aşağıdaki örnek kodda herhangi bir seçim yapılmadan birinci karta oturum açılmaktadır. *SmartCard* sınıfının *getSlotInfo(Long slot)* fonksiyonu ile slot hakkındaki bilgiler edinelebilir, burdaki slot tanımlamasıyla kullanıcıdan kart seçmesi istenebilir.

Akıllı kart ile işlem yapmaya başlamak için *openSession()* fonksiyonu ile oturum açılmalıdır. Karttan sertifika okumak için *login* olmaya gerek yoktur. Yalnız imza çözme veya ifreleme işlemi yapılacaksa karta *login* olunmalıdır.

Java5 ve C# :

```
SmartCard sc = new SmartCard(CardType.AKIS);
long [] slots = sc.getSlotList();
//sc.getSlotInfo(slots[0]).slotDescription;
long session = sc.openSession(slots[0]);
sc.login(session, "12345");
```

### 3.1 Akis Kartlara Erişim

Akis kartlara Java 6 kullanıldığında Akis'in java kütüphanesi kullanılarak komut (APDU) gönderilebilmektedir. Sistemde akis sürücüsü yüklü olmasa bile AkisCIF.x.x.x.jar olduğunda karta erişilmektedir. AkisCIF üzerinden akıllı karta erişmek akıllı kart işlemlerinin süresini dolayısıyla imza süresini kısaltmaktadır. Yalnız AkisCIF üzerinden karta erişildiğinde diğer programlar karta erişememektedir. Yeni bir sürüm akis kart kullanmaya başladığınızda AkisCIF'i de yenilemeniz gerekecektir.

Akıllı karta APDU komutları ile AkisCIF.x.x.x.jar üzerinden erişilmesinden APDUSmartCard sınıfı sorumludur. Örnek bir kullanım aşağıdaki gibidir.

```
APDUSmartCard sc = new APDUSmartCard();
long [] slots = sc.getSlotList();
sc.openSession(slots[0]);
List<byte []> certs = sc.getSignatureCertificates();
CertificateFactory cf = CertificateFactory.getInstance("X.509");
X509Certificate cert = (X509Certificate)cf.generateCertificate(new
ByteArrayInputStream(certs.get(0)));

BaseSigner signer = sc.getSigner(cert, Algorithms.SIGNATURE_RSA_SHA1);
```

Yukarıda da belirtildiği gibi AkisCIF arayüzü bütün kartları desteklemeyebilir. AkisCIF desteklendiğinde AkisCIF ile desteklenmediğinde dll ile işlemlerinizi yapmak için aşağıdaki şekilde kullanabilirsiniz.

```

BaseSmartCard bsc = null;
int index = 0;
String [] terminals = SmartOp.getCardTerminals();
String selectedTerminal = terminals[index];
long slot;
if(APDUSmartCard.isSupported(selectedTerminal))
{
    bsc = new APDUSmartCard();
    slot = index + 1;
}
else
{
    Pair<Long, CardType> slotAndCardType = SmartOp.getSlotAndCardType
(selectedTerminal);
    slot = slotAndCardType.getObject1();
    bsc = new P11SmartCard(slotAndCardType.getObject2());
}
bsc.openSession(slot);

```

Akis .NET için herhangi bir komut kütüphanesi sağlamadığından APDU komutlarının kullanımı için .NET ESYA API içerisinde bir destek bulunmamaktadır.

## 4 Akıllı Kartın Sertifikanın Okunması

Akıllı karttan sertifika *SmartCard* sınıfının *getSignatureCertificates()* veya *getEncryptionCertificates()* fonksiyonları ile okunabilir. Eğer imzalama işlemi yapılacaksa *getSignatureCertificates()* fonksiyonu, ifreleme işlemi yapılacaksa *getEncryptionCertificates()* fonksiyonu kullanılmalıdır.

Bu fonksiyonlar sertifikaların kodlanmış hallerini *byte []* olarak dönerler. Eğer ESYA API *asn* modülünü (*ma3api-asn-....jar / ma3api-asn.dll*) kullanabiliyorsanız, karttan aldığınız byte değerlerini anlamlı hale getirmek için *ECertificate* sınıfını kullanabilirsiniz.

Atılacak imzanın kanuni hükümlülüklerinin olması için imzalamada kullanılan sertifikanın nitelikli olması gerekmektedir. Bu kontrol *ECertificate* sınıfının *isQualifiedCertificate()* fonksiyonu ile yapılabilir.

*ECertificate* sınıfının *getSubject().stringValue()* fonksiyonu ile sertifikalar birbirinden ayırt edilebilir. Kullanıcı bu bilgi ile seçim yapabilir.

Ayrıca *ECertificate* sınıfının *getSubject().getCommonNameAttribute()* fonksiyonu sertifika sahibinin ismini dönmektedir. Karttaki sertifikaların isim bilgisi hepsi için aynı olacağından, karttaki sertifikaları ayırt etmek amacıyla kullanılamaz. Kimin imzayı attığını göstermek için kullanılabilir.

Aşağıdaki kod bloğu akıllı kart içinden imzalama sertifikalarını alıp nitelikli olanların *Subject* alanını ekrana basılmaktadır.  
Java :

```
List<byte []> certs = smartCard.getSignatureCertificates(session);
for (byte[] bs : certs) {
    ECertificate cert = new ECertificate(bs);
    if(cert.isQualifiedCertificate())
        System.out.println(cert.getSubject().stringValue());
}
```

**C# :**

```
List<byte []> certs = smartCard.getSignatureCertificates(session);
foreach (byte[] bs in certs) {
    ECertificate cert = new ECertificate(bs);
    if(cert.isQualifiedCertificate())
        Console.WriteLine(cert.getSubject().stringValue());
}
```

E er ESYA API asn sınıflarına eri im yoksa, *ECertificate* yerine java'nın *x509Certificate* sınıfı kullanılabilir. *ECertificate* sınıfının *isQualifiedCertificate()* fonksiyonu yerine a a ıdaki örnek kodda gösterildi i gibi kontrol yapılabilir. Sertifikaları birbirinden ayırt etmek amacıyla *x509Certificate* sınıfının *getSubjectDN().toString()* metodu kullanılabilir.

A a ıdaki kod blo u akıllı kart içinden imzalama sertifikalarını alıp nitelikli olanların *Subject* alanını ekrana basmaktadır.

```
List<byte []> certs = smartCard.getSignatureCertificates(session);
CertificateFactory cf = CertificateFactory.getInstance("X.509");
String qcStatement = "1.3.6.1.5.5.7.1.3";
for (byte[] bs : certs) {
    X509Certificate cert = (X509Certificate)cf.generateCertificate(new
    ByteArrayInputStream(bs));
    if( cert.getExtensionValue(qcStatement) != null)
        System.out.println(cert.getSubjectDN().toString());
}
```

## 5 Akıllı Karttaki Nesne Adlarının Okunması

Akıllı kartta bulunan sertifika, açık anahtar ve özel anahtarın her biri nesne olarak adlandırılır. Akıllı karttaki nesnelerin adı ile de i lem yapılabilir. Nesne adları de i ken olabilece inden nesne adları ile i lem yapmak önerilmez. Yalnız bazı durumlarda nesne adları kullanıcıya daha anlamlı gelebilir.

*SmartCard* sınıfının *getSignatureKeyLabels(...)* ve *getEncryptionKeyLabels(...)* fonksiyonları ile anahtarların adları okunabilir. E er anahtarın sertifikasının adı, anahtar adı ile aynı ise bu ad ile sertifika da okunabiir. Sertifikanın okunması için *readCertificate(long aSessionID,String aLabel)* fonksiyonu kullanılabilir.

## 6 Akıllı Karta mızalama- ifreleme lemlerinin Yapılması

Akıllı kartta ifreleme ve imzalama i lemlerinin yapılması için *login* olunması gerekmektedir. *SmartCard* sınıfının *decryptDataWithCertSerialNo(...)*, *decryptData(...)*, *signDataWithCertSerialNo(...)*, *signData(...)* fonksiyonları kriptografik i lemleri yerine getirmek için kullanılabilir. Akıllı kart ile yapılacak i lemler, özel anahtar (private key) ile yapılacak i lemler olmalıdır. Açık anahtar ile yapılan i lemlerin herhangi bir güvenlik kısıtı olmadığından akıllı kartta yapılmasının anlamı yok. Özel anahtar kullanan i lemler ise imza atma ve ifrelenmiş verinin ifresinin çözülmesi i lemidir.

Yalnız imzalama ve ifreleme i lemlerini kullanan modüller *BaseSigner* veya *BaseCipher* arayüzünde imzacılar ve ifreleyiciler istemektedir. Bu yüzden *SCSignerWithCertSerialNo*, *SCSignerWithKeyLabel*, *SCCipherWithCertSerialNo*, *SCCipherWithKeyLabel* sınıfları daha çok kullanılacaktır.

A a daki örnek kodda sertifika seri numarası ile i lem yapan sınıflar vardır.

Java:

```
SCSignerWithCertSerialNo signer = new SCSignerWithCertSerialNo(sc,
    session, slot,
        signatureCert.getSerialNumber().toByteArray(), Algorithms.
SIGNATURE_RSA_SHA1);

SCCipherWithCertSerialNo cipher = new SCCipherWithCertSerialNo(sc,
    session,
        encCert.getSerialNumber().toByteArray());
```

C#:

```
SCSignerWithCertSerialNo signer = new SCSignerWithCertSerialNo(sc,
    session, slot,
        signatureCert.getSerialNumber().GetData(), SignatureAlg.RSA_SHA1.
getName());

SCCipherWithCertSerialNo cipher = new SCCipherWithCertSerialNo(sc,
    session,
        encCert.getSerialNumber().GetData());
```

A a daki örnek kodda anahtar adı ile i lem yapan sınıflar vardır.

Java :

```
SCSignerWithKeyLabel signer = new SCSignerWithKeyLabel(sc, session, slot,
    "yasemin.akturk#SIGN0",
        Algorithms.SIGNATURE_RSA_SHA1);

SCCipherWithKeyLabel cipher = new SCCipherWithKeyLabel(sc, session, slot,
    "yasemin.akturk#ENC0");
```

C#

```
SCSignerWithKeyLabel signer = new SCSignerWithKeyLabel(sc, session, slot,
    "yasemin.akturk#SIGN0",
```

```
SignatureAlg.RSA_SHA1.getName());

SCCipherWithKeyLabel cipher = new SCCipherWithKeyLabel(sc, session, slot,
"yasemin.akturk#ENCRO");
```

## 7 PKCS7 Yapısında mza Atılması

PKCS7 yapısı en basit imza yapılarından biridir. *PKCS7Signature* sınıfı PKCS7 formatında imza atılmasından sorumlu sınıftır. Ayırık imza veya bütünle ik imza atılabilir. *signExternalContent* fonksiyonu ile ayırık imza, *signInternalContent* fonksiyonu ile bütünle ik imza atılabilir.

Aşağıdaki örnek kodda PKCS7 yapısında imzanın nasıl atılacağı gösterilmiştir. Örnekte ayırık imza atılmıştır. *signInternalContent* fonksiyonu kullanılırsa bütünle ik imza atılacaktır. Bütünle ik imzadan içerik PKCS7 nesnesinin *getContentInfo().getContentBytes()* fonksiyonu ile alınabilir.

Aşağıdaki örnek kod sadece Java için geçerli olup, PKCS7 yapısının .NET ESYA API tarafında desteği bulunmamaktadır.

```
public void testPKCS7() throws Exception
{
    PKCS7Signature pkcsSignature = new PKCS7Signature();
    ByteArrayOutputStream signature = new ByteArrayOutputStream();

    SmartCard sc = new SmartCard(CardType.AKIS);
    long [] slots = sc.getSlotList();
    //sc.getSlotInfo(slots[0]).slotDescription;
    long session = sc.openSession(slots[0]);
    sc.login(session, "12345");

    //Gets first certificate, it must be asked to user if it is more
    than one certificate.
    byte [] certBytes = sc.getSignatureCertificates(session).get(0);
    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    X509Certificate cert = (X509Certificate)cf.generateCertificate(new
    ByteArrayInputStream(certBytes));

    BaseSigner signer = new SCSSignerWithCertSerialNo(sc, session, slots
    [0], cert.getSerialNumber().toByteArray(),
    Algorithms.SIGNATURE_RSA_SHA1);

    ByteArrayInputStream bais = new ByteArrayInputStream(toBeSigned);
    pkcsSignature.signExternalContent(bais, cert, signature, signer);

    Assert.assertEquals(true, validate(new ByteArrayInputStream
    (signature.toByteArray()), cert));
```



```
}  
  
//For this case, there is one signature  
private boolean validate(InputStream signature, X509Certificate cert)  
throws Exception  
{  
    PKCS7 p = new PKCS7(signature);  
    //validates the signature, not the person.  
    SignerInfo [] signerInfo = p.verify(toBeSigned);  
    if(signerInfo == null)  
        return false;  
    else  
    {  
        //Checks whether the expected person signed the data.  
        return signerInfo[0].getCertificateSerialNumber().equals(cert.  
getSerialNumber()) == true;  
    }  
}  
  
private final static byte [] toBeSigned = "Test".getBytes();
```

## 8 Lisans Ayarları

*LicenseUtil.setLicenseXml(InputStream ...)* fonksiyonu ile lisans dosyası verilebilir. Eğer bu fonksiyon ile bir lisans bilgisi verilmemişse, 'working directory' altında lisans klasörü altında "lisans.xml" dosyası aranır.

Lisans dosyasının satıcı yapılan mü teri dı ında ba kasının eline geçmemesi için bu dosyanın sunucuda tutulması, istemci yazılımlarının lisansa sunucudan erişmesi tavsiye edilmektedir.

### 8.1 Test Lisansı ile Çalışma

ESYA API ile denemeler yapmanız için test lisansı edinebilirsiniz. Lisans için verilen xml dosyasında, test alınının de eri "test" ise deneme lisansına sahipsinizdir.

Bu lisans ile yaptığımız çalışmalarda ancak "common name" alanında "test" metni geçen sertifikalar kullanabilirsiniz. Ayrıca akıllı kart ile yapacağınız işlemlerde birkaç saniye gecikme olacaktır.