Kullanım Kılavuzu

Cybersoft Enformasyon Teknolojileri Ltd. Şti.
ODTU Teknokent Silikon Binaları 1. Kat No: 18 06531
ODTU / Ankara
Türkiye
Tel : +90 312 210 19 40

Giriş

CSSigner, Cybersoft firması tarafından geliştirilmiş açık kaynak kodlu e-imza yazılım kütüphanesidir.

CSSigner API kullanarak;

- herhangi bir dosya veya karakter dizisini kolay bir şekilde imzalayabilir,
- imzalı bir dosya ya da karakter dizisinin imzasının sağlamasını yapabilir (verification), imzalayan sertifikaların geçerliliğini (validation) test edebilir
- imzaya konu olan bileşenleri (imzalanan içerik, imzalayan sertifika, SİL listeleri) ayrıştırabilirsiniz.

CSSigner API kullanarak paralel ve seri imza atabilir, imzalara zaman damgası ekleyebilirsiniz.

CSSigner yazılımı içerisine eklenmiş olan kullanıcı arayüzüne sahip CSSignerApp modülü ile CSSigner API'sini kullanmadan imza ile ilgili işlemlerinizi kolayca yapabilirsiniz.

CSSigner büyük dosyaların imzalanabilmesi için optimize edilmiştir.

Teknik Özellikler

- Java (JDK1.5) programlama dili ile geliştirilmiştir.
- ETSI TS 101 733 elektronik imza formatına uygundur.
- X.509 formatında Nitelikli Sertifikalar ile çalışır (RFC 2459).
- PKCS #11 arayüz standardı sağlayan kartlarla çalışır.
- Paralel ve seri imza atılabilir.
- İmzalara zaman damgası eklenebilir.
- İmzalanan dosyaların çok büyük olmaları durumunda bellek sorunu oluşmaması için disk üzerinde geçici alanlar kullanılarak imzalama yapılır.
- Platform bağımlılığı yoktur. Windows, Unix ve Linux işletim sistemlerinde kullanılabilir.
- RFC 3161, RFC 3126, RFC 2560, RFC 2634, RFC 2630, RFC 3280, RFC 3369 yayınlarında tanımlanan yapılar baz alınarak geliştirilmiştir.

CSSigner API Kullanım

Programlarınızın içerisinden CSSigner API aracılığı ile imzalama işlemini gerçekleştirebilirsiniz. CSSigner, Java programlama dili ile geliştirilmiştir. Java programlarından kolay bir şekilde kullanılabilir. Yapılması gereken işlem CSSigner.jar dosyasını proje classpath'ine eklemektir.

Örnekler:

Örnek 1 : Kart/Token Test Programı

Bu program ile elinizde bulunan bir kart/token üzerinde yer alan sertifikaları görebilirsiniz. Aynı zamanda PKCS11 kütüphanesi test edilmiş olur. Programı çalıştırmak için

java –cp CSSigner.jar ornek.KartTest

```
//-----------------
package ornek;
import java.security.KeyStore;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;

import tr.com.cs.signer.cert.C_Certificate;
import tr.com.cs.signer.cert.C_KeyStore;

public class KartTest
{
 public static void main(String[] args)
 {
    if (args.length != 2)
    {
       System.err.println("usage : KartTest <pkcs11 dll/so name> <password>");
       System.exit(-1);
    }
    try
    {
       char[] password = args[1].toCharArray();
       KeyStore keyStore = C_KeyStore.createKeyStore("PKCS11", "DENEME", args[0], password);
       System.out.println("Kartta Bulunan Sertifikalar");
       System.out.println("=========================");
       Map<String, C_Certificate> certs = C_KeyStore.getCerts(keyStore);
       Iterator<Entry<String, C_Certificate>> iterator = certs.entrySet().iterator();
       while (iterator.hasNext())
       {
          Entry<String, C_Certificate> entry = iterator.next();
          System.out.println("Sertifika Sahibi : " + entry.getValue().getSubjectName());
       }
    }
    catch (Exception e)
    {
       e.printStackTrace();
    }
 }
}
//-----------------
```

Örnek 2 : Kart/Token Test Programı [Grafik arayüz]

Bu program ile elinizde bulunan bir kart/token üzerinde yer alan sertifikaları görebilirsiniz. Aynı zamanda PKCS11 kütüphanesi test edilmiş olur. Programı çalıştırmak için

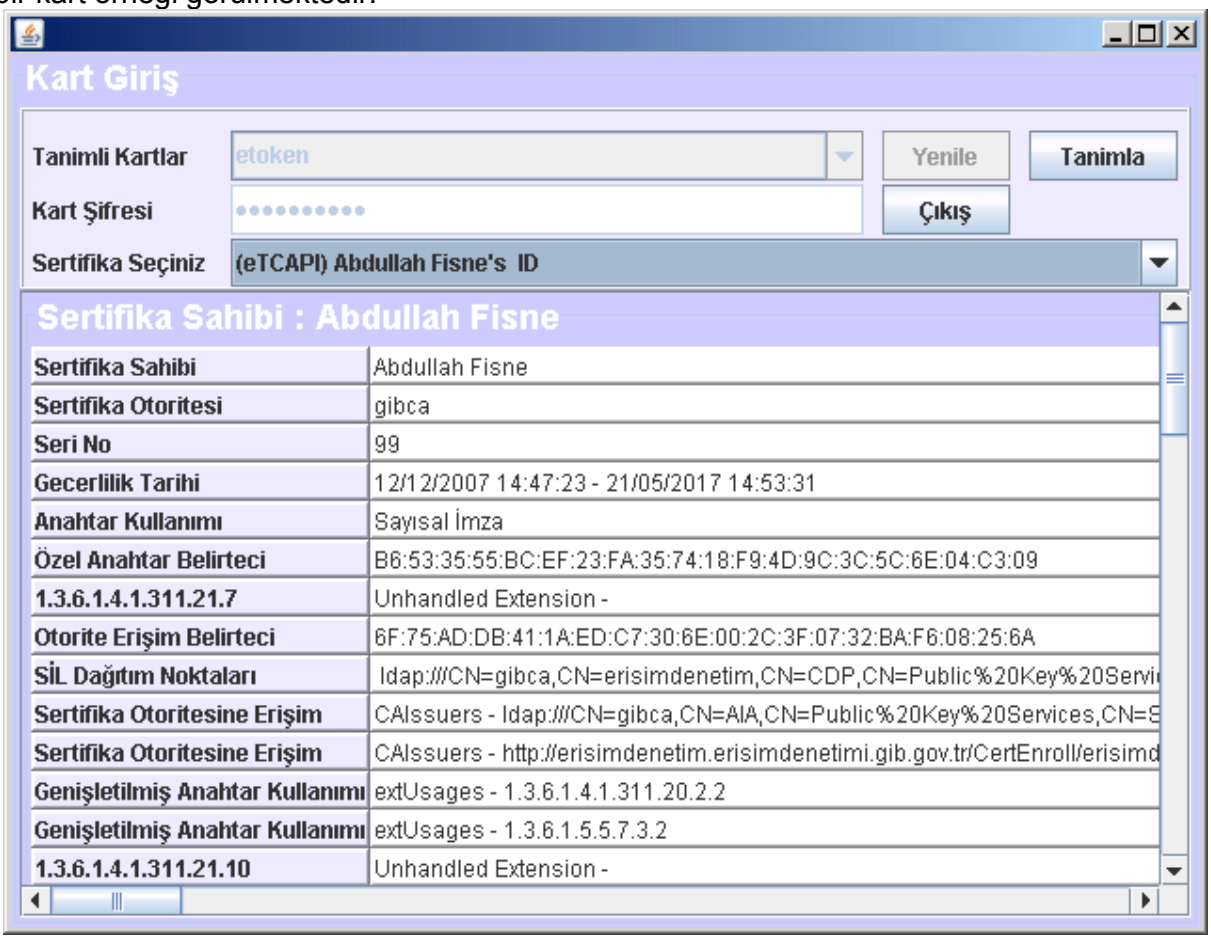
java –cp CSSigner.jar ornek.KartTestGUI

```
//------------
package ornek;
import javax.swing.JFrame;

import tr.com.cs.signer.cert.C_CardLogin;
import tr.com.cs.signer.swing.CSSignerApp;

public class KartTestGUI
{
   public static void main(String[] args) throws Exception
   {
      C_CardLogin crdLogin = new C_CardLogin();
      JFrame frame = new JFrame();
      frame.setContentPane(crdLogin);
      frame.setSize(640, 480);
      CSSignerApp.placeToCenter(frame);
      frame.setVisible(true);
   }
}
//------------
```

Program çalıştığı zaman ekrana Şekil 1' deki gibi bir görüntü gelecektir. Kullanıcı ana dizininde yer alan cssign.properties adlı özellik dosyasında yer alan kart tanımlarına göre kart bilgilerini görmek için kullanıcıdan şifre isteyecektir. Şifre girildiği zaman kart içerisinde bulunan sertifikalar görüntülenir. Yeni bir kart tanımlamak için tanımla düğmesine basılıp kartın adı ve PKCS11 kütüphanesinin verilmesi yeterlidir. Aşağıda Şekil 1'de giriş yapılan bir kart örneği görülmektedir.



Şekil 1 : Kart Bilgilerini Görme

Ekranda **Tanımla** düğmesine basıldığı zaman Şekil 2 'deki ekran gelecek ve kullanıcıdan kart bilgileri istenecektir. Burada girilen bilgiler cssign.properties adlı dosyaya kaydedilmekte ve CSSignerApp kullanıldığı zaman kart bilgileri içerisinde seçimi sağlanmaktadır.

Şekil 2 : Yeni Kart Tanımlama

Örnek 3 : Dosya İmzalama

Aşağıda verilen program parçası kullanıcıdan kart, PKCS11 kütüphane dosyası, parola ve dosya adını alır. Kartta bulunan sertifikalardan imzalama yapılmak istenen sertifika ile dosyayı imzalar ve dosya adına .imz ekleyerek imzalı dosyayı saklar.

```
//-----------
package ornek;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.security.KeyStore;
import java.security.PrivateKey;
import java.util.Iterator;
import java.util.List;

import tr.com.cs.signer.cert.C_Certificate;
import tr.com.cs.signer.cert.C_KeyStore;
import tr.com.cs.signer.cms.C_FileSigner;

public class DosyaImzala
{
 public static void main(String[] args) throws Exception
 {
    if (args.length != 3)
    {
       System.err.println("usage : DosyaImzala <pkcs11 dll/so name> <parola> <dosya adi>");
       System.exit(-1);
    }
    char[] password = args[1].toCharArray();
    String tbsFile = args[2];
    KeyStore keyStore = C_KeyStore.createKeyStore("PKCS11", "DENEME", args[0], password);
    System.out.println("Kartta Bulunan Sertifikalar");
    System.out.println("==========================");
    List<C_Certificate> certsList = C_KeyStore.certificates(keyStore);
    Iterator<C_Certificate> iterator = certsList.iterator();
    int certNo = 0;
    while (iterator.hasNext())
    {
       System.out.println("Sertifika [" + ++certNo + "] ->" + iterator.next().getSubjectName());
    }
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
```

```
      int cNo = 1;
      while (true)
      {
         System.out.print("Imzalama Yapmak Istediginiz Sertifika No : ");
         String readLine = in.readLine();
         try
         {
            cNo = Integer.valueOf(readLine).intValue();
         }
         catch (NumberFormatException e)
         {
            e.printStackTrace();
            continue;
         }
         if (cNo < 1 || cNo > certsList.size())
         {
            System.out.println("0 < Sertifika No < " + certsList.size() + " Olmalidir!..");
            continue;
         }
         break;
      }
      String alias = C_KeyStore.getCertAliasJavaKeyStore(keyStore, cNo);//Setfifikaya ait olan alisai
al.
      C_Certificate signerCert = certsList.get(cNo - 1);
      PrivateKey pkey = (PrivateKey) keyStore.getKey(alias, password);
      System.out.println("Dosya " + signerCert.getSubjectName() + " Sertifikasi Ile
Imzalanacaktir...");
      C_FileSigner fileSigner = new C_FileSigner(tbsFile);
      fileSigner.sign(keyStore.getProvider(), signerCert, pkey, null);
      fileSigner.save(tbsFile + ".imz");
      System.out.println("Imzali Dosya : " + tbsFile + ".imz");
   }
  }
```

Örnek 4 : İmzalı Dosya Doğrulama

İmzalanmış bir dosya aşağıda verilen program ile ayrıştırılıp imzalanan kısım ve imzalayan sertifikalar bulunabilir. Program iki parametre alır. Birinci parametre imzalı dosya adı, ikinci parametre ise imzalanmış içeriğin yazılacağı dosyanın adıdır.

java –cp CSSigner.jar ornek.KartTestGUI XX.imz XX.data

```
//-----------
package ornek;
import java.util.Collection;

import tr.com.cs.signer.cert.C_Certificate;
import tr.com.cs.signer.cms.C_SignedData;
import tr.com.cs.signer.cms.C_SignerInfo;
import tr.com.cs.signer.cms.C_Verifier;

public class ImzaDogrula
{
 public static void main(String[] args) throws Exception
 {
    if (args.length != 2)
    {
       System.err.println("usage : ImzaDogrula <imzali dosya adi> <veri dosya
adi>");
       System.exit(-1);
    }
    C_SignedData signedData = C_Verifier.verify(args[0], args[1]);
    Collection<C_SignerInfo> signerInfos = signedData.getSignerInfos();
    System.out.println("Imzalayan Sertifikalar");
    System.out.println("======================");
    for (C_SignerInfo signerInfo : signerInfos)
    {
       C_Certificate signingCert = signerInfo.getSigningCert();
       System.out.println(signingCert.getSubjectName());
    }
 }
}
//-----------
```

Buraya kadar verilmiş olan küçük program parçaları API kullanımı hakkında kısa bir fikir vermektedir. Yazılımda Kullanılan Yapılar ve Önemli Sınıflar bölümünde ileri düzey (paralel ve seri imza, imzalara zaman damgası ekleme, imzaların imzalı ve imzasız niteliklerinin alınması gibi) imzalama işlemleri için kullanılan sınıflar hakkında bilgi verilecektir.

CSSignerApp Programı

CSSignerApp; grafik ortamda çalışan ve basit düzeyden ileri düzeye kadar bütün imza işlemlerinin yapılabildiği bir programdır. Bu program yardımı ile imzalama işlemleri yanında sertifika ile ilgili işlemler (doğrulama, sertifikasyon zincirini alma vs), imzalı dosyanın ayrıştırılması ve doğrulanması, SİL listelerinin görüntülenmesi, kart/token tanımlama ve içerisinde olan sertifikaların görüntülenmesi gibi işlemler de kolay bir şekilde yapılabilir.

Programı çalıştırmak için

- **java –jar CSSigner.jar**
- **java –cp CSSigner.jar tr.com.cs.signer.swing.CSSignerApp**

komutlarından birisini yazmanız yeterlidir. İlk olarak kullanıcıdan internete çıkış için proxy kullanıp kullanmadığı sorulur. Sertifika doğrulamaları ya da zaman damgası alma işleminde internet bağlantısı için kullanılacak olan bu değerler aynı zamanda cssign.properties dosyasına yazılır. Daha sonraki çalışmalarda bu değerler dosyadan okunur ve kullanıcı değiştirmek istemiyor ise sadece onaylayıp geçmesi yeterlidir. Eğer internete direk çıkış var ise proxy kullanımının *false* olarak set edilmesi yeterlidir.

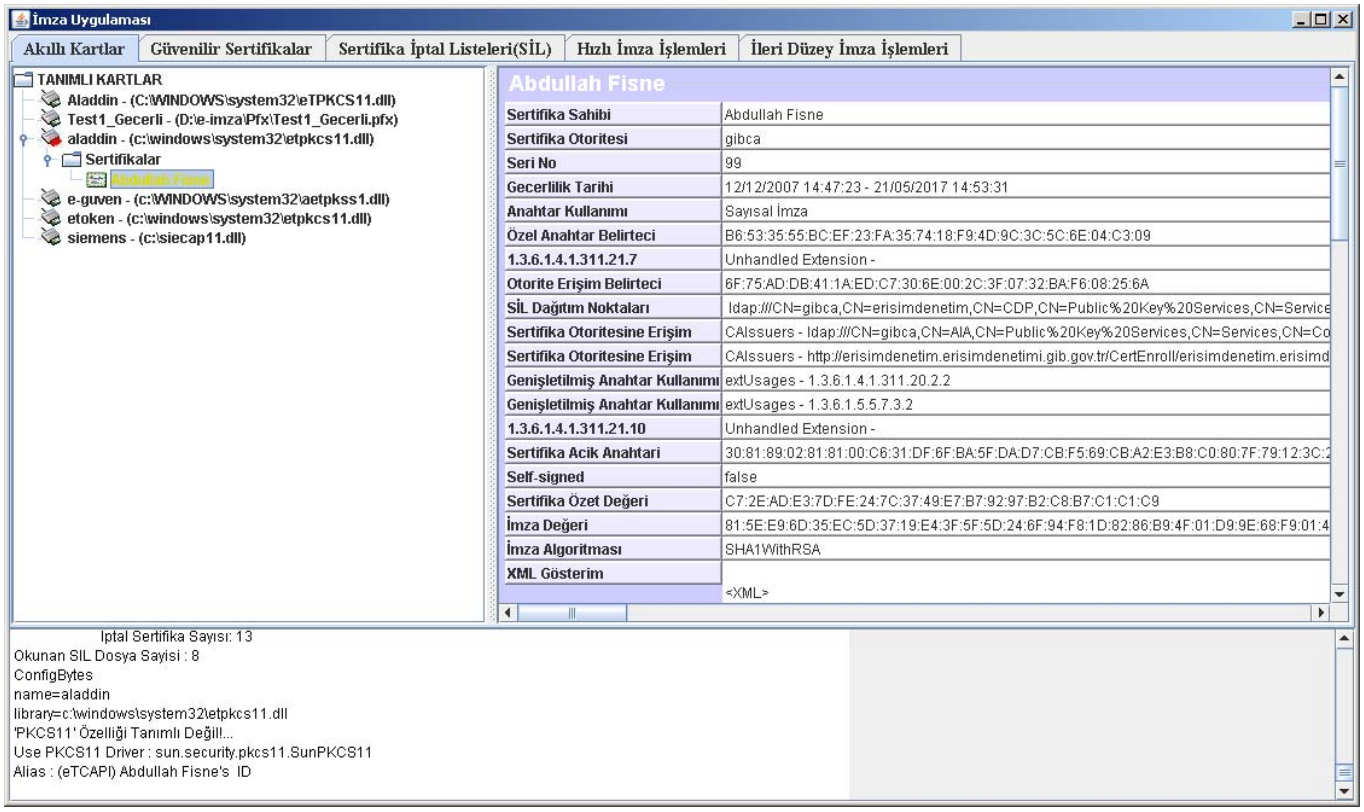
Şekil 3: Proxy (Vekil Sunucu) Kullanım Ekranı

Proxy Kullanımı ekranından sonra ana ekran gelecektir. Ana ekran Akıllı Kartlar, Güvenilir Sertifikalar, Sertifika İptal Listeleri, Hızlı İmza İşlemleri ve İleri Düzey İmza İşlemleri bölümlerinden oluşmaktadır. Aşağıda bu bölümlerle ilgili kısa bilgiler verilecektir.

Akıllı Kartlar

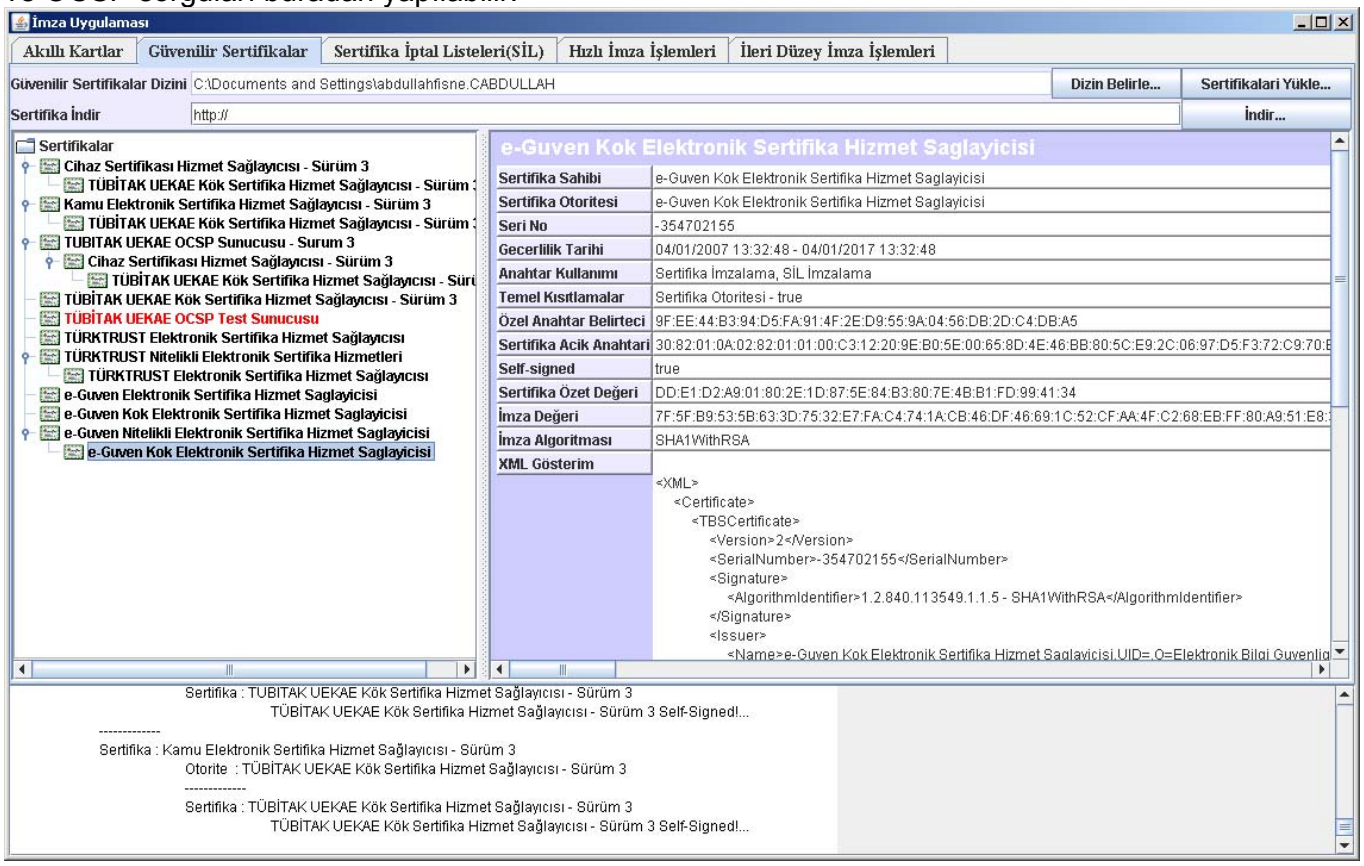
Bu bölümde akıllı kart/token tanımı yapılabilir, tanımlanmış kart silinebilir ve kart içerisindeki sertifikalar görülebilir. Kartların CSSignerApp tarafından görülebilmesi için öncelikle sisteme kurulması gerekmektedir. Kart/token ile verilen yazılımlar aracılığı ile bu kurulumun yapılmış olması gerekmektedir. Burada yapılan işlem; ilgili kartın PKCS11 standartları kapsamında iletişim kurabilmesini sağlayabilmek için üretici firma tarafından verilen kütüphane aracılığı ile kartla iletişim kurabilmektir.

Program genelinde herhangi bir menü parçası üzerinde farenin sağ tuşu ile yapılabilecek işlemler tanımlanmıştır.



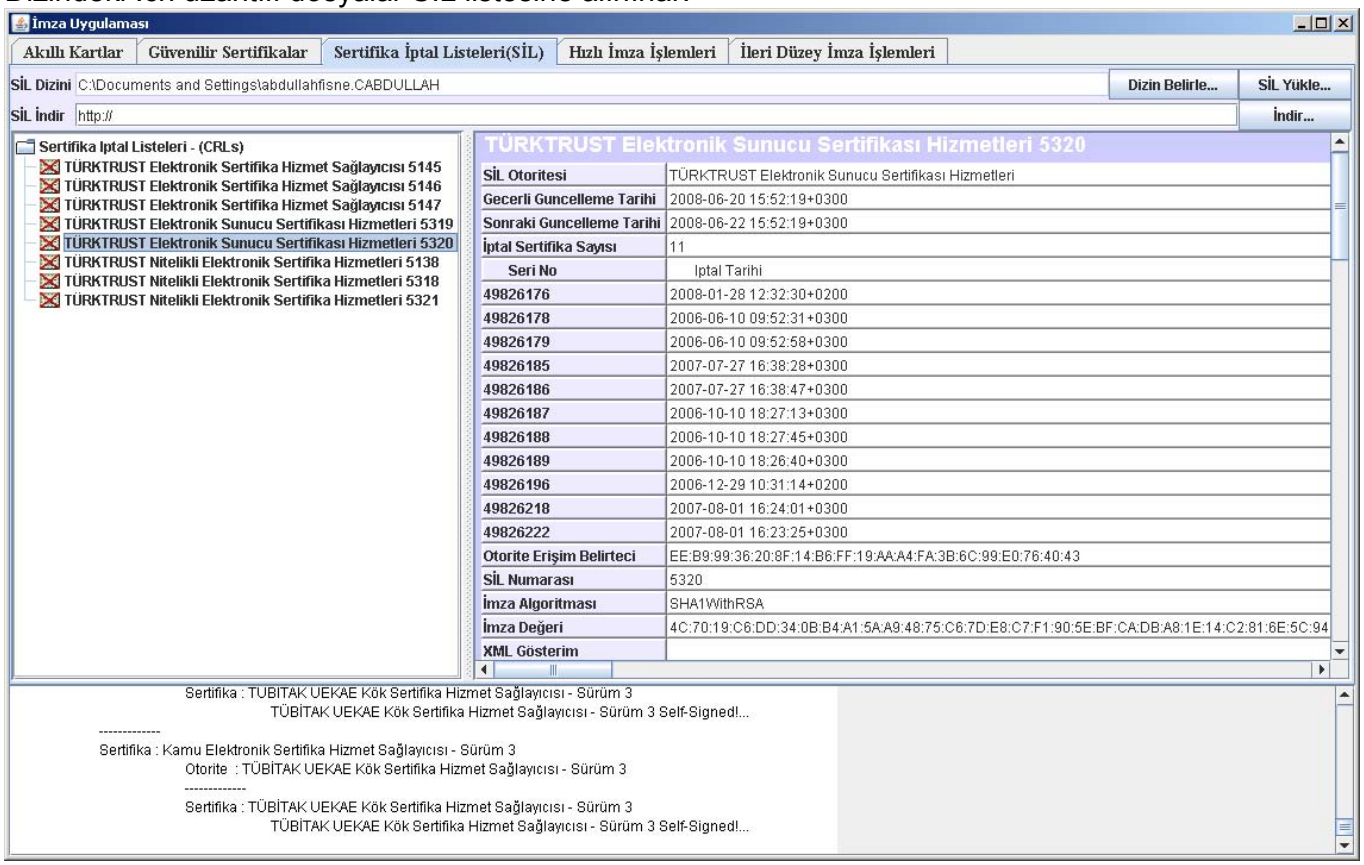
Güvenilir Sertifikalar

CSSigner.jar içerisinde /certificates dizininde tanımlı güvenilir sertifikalar yer almaktadır. Aynı zamanda kullanıcı, bu sertifikaların dışında başka güvendiği sertifikalar var ise bu sertifikaların bulunduğu dizini belirleyip buradaki sertifikaların da güvenilir sertifikalar içerisine alınmasını sağlayabilir. Dizindeki sertifika dosyalarının .cer veya .crt uzantılı olması yeterlidir. Internet üzerinden herhangi bir adresten bir sertifika alınıp disk ortamına kaydedilebilir. Sertifikasyon zinciri ve OCSP sorguları buradan yapılabilir.



Sertifika İptal Listeleri(SİL)

CSSigner.jar içerisinde /crls dizininde tanımlı SİL listeleri yer almaktadır. Kullanıcı herhangi bir SİL adresinden SİL listesi alabilir, bunları kendi belirlediği bir dizine saklayıp sisteme tanıtabilir. Dizindeki .crl uzantılı dosyalar SİL listesine alınırlar.



Hızlı İmza İşlemleri

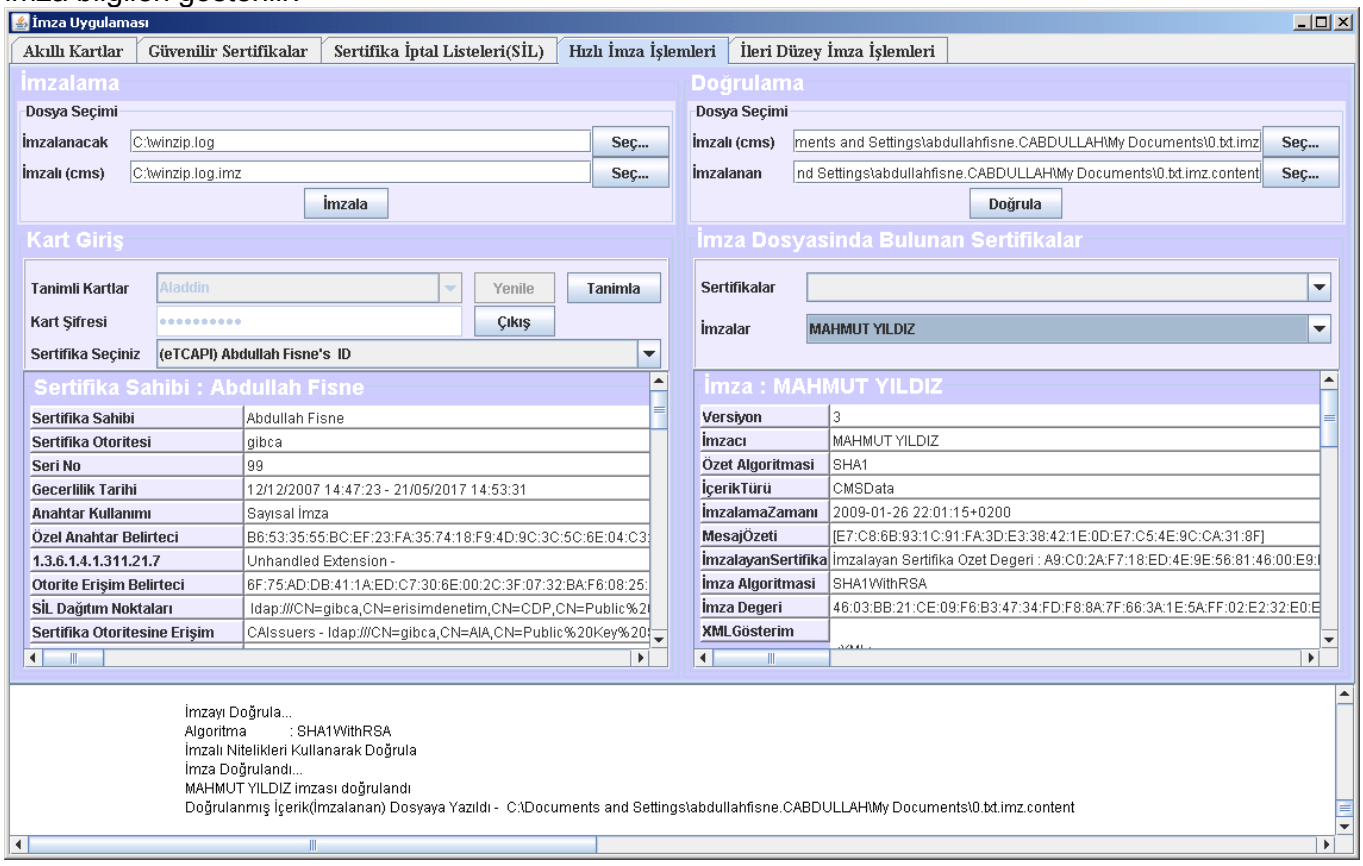
Bu bölümde imzalama ve doğrulama işlemleri hızlı bir şekilde yapılabilmektedir.

İmzalama

Sistemde akıllı kartınız takılı ve tanımlı ise imzalamak istediğiniz dosyayı *İmzalama* bölümünde *İmzalanacak* kısmına girmeniz yeterlidir. İmzalı dosya genel olarak dosya adına .imz eklentisi eklenerek oluşturulur. Değiştirmek isterseniz *İmzalı(cms)* dosya adını değiştirmeniz yeterlidir.
Daha sonra *Kart Giriş* bölümünde karta giriş yapmalısınız. Sertifika seçiminden sonra *İmzala* düğmesine basıldığı zaman imzalama işlemi gerçekleşecektir.

Doğrulama

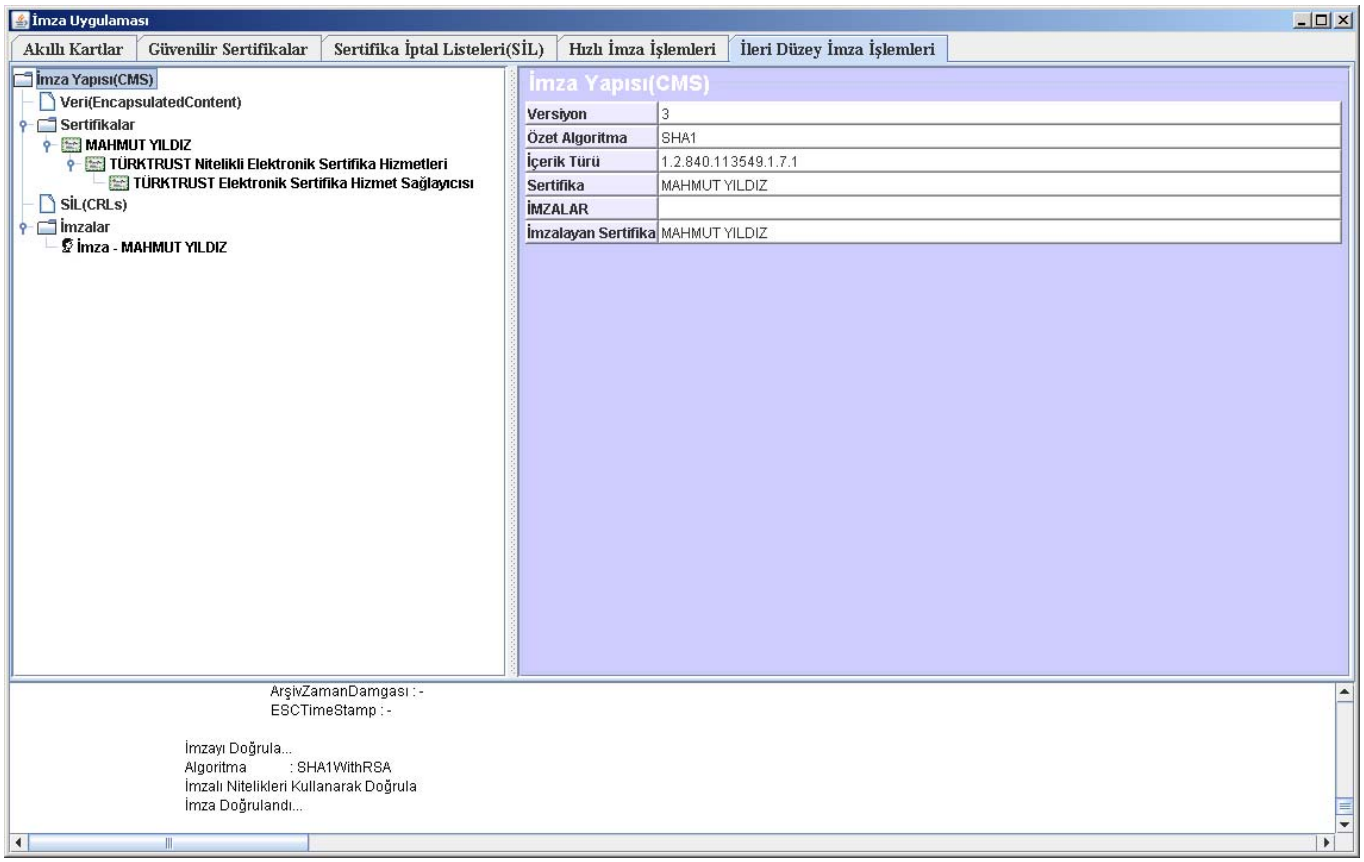
Doğrulama yapılacak kısımda *İmzalı(cms)* kısmına imzalanmış dosyanın adını girmeniz ve *Doğrula* düğmesine basmanız yeterlidir. İmzalı içeriği başka bir dosyaya saklamak isterseniz *İmzalanan* kısmını doldurmalısınız.
Bu bölümün alt kısmında imzalı dosyada bulunan sertifikalar ve imzalar ayrı ayrı görüntülenebilir. Sertifika seçimi yapılmış ise alt kısımdaki tabloda sertifika bilgileri, imza seçimi yapılmış ise ilgili imza bilgileri gösterilir.



İleri Düzey İmza İşlemleri

İmzalı bir dosyayı incelemek, paralel ya da seri imza atmak, zaman damgası eklemek gibi bir çok özellik içermektedir. Burada her işlem manuel olarak yapılabilir. Menü üzerinde sağ tuş yapılabilecek işlemleri göstermektedir. Eğer imzalı doküman incelenecek ise *İmza Yapısı(CMS)* üzerinde sağ tuş ile imzalı dosyanın açılması gerekir. İmzalı dosya açıldığı zaman gerekli alt menü detayları otomatik olarak doldurulur. Bundan sonra paralel/seri imza atılabilir, zaman damgası eklenebilir. İmzalı içerik dosyaya saklanabilir.

İmzalı olmayan bir dosyayı sıfırdan imzalamak ve yukarıdaki işlemleri yapmak isterseniz *Veri(EncapsulatedContent)* üzerinde sağ tuşa basıp imzasız dosya açmanız ve imzalama işlemine aynı şekilde devam etmelisiniz. *Her iki işlem sonucunda ilgili dosyayı saklamalısınız.*



cssign.properties Dosyası

Programın kullandığı bir kaynak(resource) dosyasıdır. Kullanıcının home dizini altında bulunur. Yeni bir özellik eklendiği zaman diske tekrar yazılır. C_Properties sınıfı tarafından kullanılır. İçerisindeki değerler ve kullanım amaçları aşağıdaki gibidir.

- **certs.location** Güvenilir sertifikaların (trusted certificates) bulunduğu dizin. Bu dizinde bulunan *cer* ve *crt* uzantılı dosyalar sertifika dosyaları olarak kabul edilir ve CSSigner tarafından herhangi bir sertifikanın kök sertifikası aranırken öncelikle bu sertifikalar içerisinde var olup olmadığına bakılır.
- **crls.location** SİL listelerinin bulunduğu dizin. *crl* uzantılı dosyalar sertifika iptal listesi olarak kabul edilir. Sertifika doğrulama işleminde kullanılır. Dönem dönem bu dizine sertifika iptal listesi yayınlayan kuruluşlardan SİL listeleri alınıp eklenebilir.
- **smart.cards** CSSignerAPP uygulaması kullanıldığı zaman sisteme giriş yapılması istenirse buradaki kartlar kullanıcıya listelenir. Formatı {kartadı, kütüphane dosyası}* şeklindedir.
- **tsa.url** Zaman Damgası alınacak adres.
- **tsa.username** Zaman Damgası almak için kullanılacak kullanıcı adı.
- **tsa.password** Zaman Damgası almak için kullanılacak kullanıcı şifresi.
- **tsa.use.tubitak.api** Zaman Damgası Tübitak'tan alınacak ise ilgili kurum tarafından sağlanan API kullanılmak zorundadır. O durumda bu değer *true* olarak set edilir.
- **http.useProxy** Internet bağlantısı proxy üzerinden yapılacak ise bu değer *true* set edilir.
- **http.proxyHost** Internet bağlantısı proxy üzerinden yapılacak ise bu değer proxy sunucusunun IP adresini gösterir.
- **http.proxyPort** Internet bağlantısı proxy üzerinden yapılacak ise bu değer proxy sunucusunun port numarasını gösterir.

Örnek cssign.properties dosyası

```
//----------
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="certs.location">c:/trustedcerts</entry>
<entry key="crls.location">c:/crls</entry>
<entry key="smart.cards">kart_1,kütüphanedosyası_1,  kart_2, kütüphanedosyası_2</entry>
<entry key="http.useProxy">true</entry>
<entry key="http.proxyHost">10.251.75.10</entry>
<entry key="http.proxyPort">8080</entry>
<entry key="tsa.use.tubitak.api">true</entry>
<entry key="tsa.url">http://zd.kamusm.gov.tr</entry>
<entry key="tsa.username">XXX</entry>
<entry key="tsa.password">YYY</entry>
</properties>
//----------
```

Yazılımda Kullanılan Yapılar ve Önemli Sınıflar

/*İmza Yapısı – Bakınız RFC 3369*/

```
public class C_SignedData extends C_DEREncodable implements I_PreEncoding
{
      T_INTEGER                          version;
      Collection<C_AlgorithmIdentifier>  digestAlgorithms  = new LinkedHashSet<C_AlgorithmIdentifier>();
      C_EncapsulatedContentInfo          encapContentInfo;
      Collection<C_Certificate>          certificates;
      C_CertificateRevocationLists       crls;
      Collection<C_SignerInfo>           signerInfos       = new TreeSet<C_SignerInfo>();
}
```

/*İmzalı İçerik*/

```
public class C_EncapsulatedContentInfo extends C_DEREncodable
{
   T_OBJECTIDENTIFIER      eContentType;
   G_OCTETSTRING           eContent;
}
```

/*Sertifika Yapısı - Bakınız RFC 3280*/

```
public class C_Certificate extends C_DEREncodable implements I_Properties
{
   C_TBSCertificate        tbsCertificate;
   C_AlgorithmIdentifier   signatureAlgorithm;
   T_BITSTRING             signatureValue;
}
public class C_TBSCertificate extends C_DEREncodable implements I_Properties
{
   T_INTEGER               version;
   T_INTEGER               serialNumber;
   C_AlgorithmIdentifier   signature;
   C_Name                  issuer;
   C_Validity              validity;
   C_Name                  subject;
   C_SubjectPublicKeyInfo  subjectPublicKeyInfo;
   T_BITSTRING             issuerUniqueID;
   T_BITSTRING             subjectUniqueID;
   C_Extensions            extensions;
}
```

/*İmzacı Yapısı – Bakınız RFC 3369*/

```
public class C_SignerInfo extends C_DEREncodable implements I_Properties
{
   protected T_INTEGER                    version;
   protected C_SignerIdentifier           signerIdentifier;
   protected C_AlgorithmIdentifier        digestAlgorithm;
   protected Collection<T_Attribute>      signedAttrs;
   protected C_AlgorithmIdentifier        signatureAlgorithm;
   protected T_OCTETSTRING                signature;
   protected Collection<T_Attribute>      unSignedAttrs;
}
```

/*Algoritma Belirteci– Bakınız RFC 3369*/

```
public class C_AlgorithmIdentifier extends C_DEREncodable
{
   T_OBJECTIDENTIFIER              identifier;
   Collection<C_DERObject>         parameters;
}
```

/*Attribute Yapısı - Bakınız RFC 3369*/

```
public abstract class T_Attribute extends C_DEREncodable
{
   private T_OBJECTIDENTIFIER          attrType;
   private Collection<C_DEREncodable>  attrValues  = new
LinkedHashSet<C_DEREncodable>();
}
```

/*SİL Yapısı - Bakınız RFC 3369*/

```
public class C_CertificateRevocationLists extends C_DEREncodable
{
   Collection<C_CertificateList>  certRevLists  = new
LinkedHashSet<C_CertificateList>();
}
public class C_CertificateRevocationLists extends C_DEREncodable
{
   Collection<C_CertificateList>  certRevLists  = new
LinkedHashSet<C_CertificateList>();
}
public class C_CertificateList extends C_DEREncodable implements I_Properties
{
   C_TBSCertList      tbsCertList;
   C_AlgorithmIdentifier signatureAlgorithm;
   T_BITSTRING        signatureValue;
}
public class C_TBSCertList extends C_DEREncodable implements I_Properties
{
   T_INTEGER                  version;
   C_AlgorithmIdentifier          signature;
   C_Name                     issuer;
   T_TIME                     thisUpdate;
   T_TIME                     nextUpdate;
   Map<T_INTEGER, C_RevokedCertificate>  revokedCertificates;
   C_Extensions                 crlExtensions;
}
```