

TÜBİTAK BİLGEM

MA3 API CMS Zarf

KISALTMALAR

AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CMS	Cryptographic Message Syntax
ECDH	Elliptic Curve Diffie-Hellman
JRE	Java Runtime Environment
HSM	Hardware Security Module
KDF	Key Derivation Function
MA3	Milli Açık Anahtar Altyapısı
OAEP	Optimal Asymmetric Encryption Padding
PKCS	Public Key Cryptography Standards
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman Algorithm
SHA	Secure Hash Algorithm

İçindekiler

KISALTMALAR	1
1. Giriş	3
2. Veri Şifreleme	3
3. Şifreli Verinin Çözülmesi	4
4. Çözücüler	5
4.1 Akıllı Kart Çözücü	5
4.2 Bellekte Çözücü	5
4.3 Microsoft Sertifika Deposundan Çözücü	6
5. Şifrelenmiş Veriye Yeni Alıcı Ekleme	6
6. Dizin Sisteminden Sertifika Bulma	8
7. Sertifika Doğrulama	8
8. Donanım Tabanlı Rassal Sayı Üretme	8
8.1 Akıllı Kart Tabanlı Rassal Sayı Üretme	9
EK 1. Şifreleme API Güvenli Kullanım Tavsiyeleri	10

1. Giriş

CMS Zarf API'si, şifreli veri oluşturulmasını ve şifreli verinin çözülmesini sağlar.

CMS Envelope yapısında asimetrik ve simetrik şifreleme işlemi birlikte kullanılır. Şifrelenecek olan veri, simetrik anahtarla şifrelenir. Her alıcı için bu simetrik anahtar, asimetrik algoritmalar kullanılarak şifrelenir ve şifreli verinin sonuna eklenir. Dokümanın şifresini çözmek isteyen kişi, asimetrik algoritma ile kendisi için şifrelenmiş alanın şifresini çözer. Böylelikle dokümanın şifrelendiği simetrik anahtarı elde etmiş olur. Bu simetrik anahtar ile de dokümanın şifresini çözebilir. Bu işlemler MA3 API CMS Zarf Kütüphanesi tarafından yapılır, bu kütüphaneyi kullanan geliştiricinin bu yapılarla ilgilenmesine gerek yoktur.

Bir veriyi bir kullanıcıya şifreli olarak göndermeden önce sertifikasının geçerliliği kontrol edilmelidir. Kullanıcı akıllı kartını çaldırmış ve sertifikasını iptal ettirmiş olabilir.

CMS Envelope modülü, internet üzerinden dağıtımı yapılan E-İmza Kütüphanesi paket içeriğinde bulunmamaktadır, dağıtımı lisanslı olarak yapılmaktadır.

2. Veri Şifreleme

Bir veriyi şifrelemek için `CmsEnvelopeGenerator` veya `CmsEnvelopeStreamGenerator` sınıfları kullanılabilir. Büyük verilerin şifrelenmesi için `CmsEnvelopeStreamGenerator` sınıfı kullanılmalıdır. `CmsEnvelopeGenerator` sınıfı bellek üzerindeki bir veriyi şifrelediğinden, şifreleyebileceği dosyanın boyutu .Net Framework veya JRE'nin bellek sınırı kadardır.

CMS Envelope yapısının ayarları için `EnvelopeConfig` sınıfı kullanılmalıdır. Bu sınıf yardımıyla varsayılan anahtar taşıma (`RSA_OAEP_SHA256`) ve anahtar anlaşma (`ECDH_SHA256KDF`) algoritmaları değiştirilebilir veya simetrik şifreleme algoritması değiştirilebilir.

Bir doküman bir kişiye veya daha fazla kişiye şifrelenebilir. Eğer şifrelenmiş dokümanın şifresi çözülebiliyorsa, bu dokümana yeni alıcılar eklenebilir.

Şifreli veri göndermek için, verinin gönderileceği kişi veya kişilerin sertifikalarının elimizde ve geçerli olması gerekmektedir. İptal edilmiş bir sertifika için şifreleme yapılmaması gerektiğinden her bir alıcı sertifikası için kütüphane sertifika doğrulama yapar. `addRecipients` metodu, parametre olarak alıcının sertifikasını alır ve bu sertifikayı doğrular; sertifika geçerli ise alıcı için anahtar bilgisini dokümana ekler.

Java

```
EnvelopeConfig config = new EnvelopeConfig ();
config.setPolicy(TestConstants.getPolicyFile());

CmsEnvelopeStreamGenerator cmsGenerator = new
CmsEnvelopeStreamGenerator(plainInputStream, CipherAlg.AES256_CBC);
cmsGenerator.addRecipients(config, recipientCert);
cmsGenerator.generate(encryptedOutputStream);
```

C#

```
EnvelopeConfig config = new EnvelopeConfig();
config.setPolicy(TestConstants.GetPolicyFile());
CmsEnvelopeStreamGenerator cmsGenerator = new
CmsEnvelopeStreamGenerator(plainInputStream, CipherAlg.AES256_CBC);
cmsGenerator.addRecipients(config, recipientCert);
cmsGenerator.generate(encryptedOutputStream);
```

3. Şifreli Verinin Çözülmesi

Şifreli verinin çözülmesi için CmsEnvelopeParser veya CmsEnvelopeStreamParser sınıfı kullanılabilir. Büyük şifreli verilerin çözülmesi için CmsEnvelopeStreamParser sınıfı kullanılmalıdır.

Şifrelenmiş verinin çözülmesi için verinin şifrelendiği sertifika ve o sertifikaya ait şifre çözücünün elimizde olması gerekmektedir. Şifre çözücü olarak akıllı kart veya bellek kullanılabilir.

Aşağıda akıllı kart ile şifreli veriyi çözen örnek kod bloğu verilmiştir.

Java

```
SmartCard sc = new SmartCard(CardType.AKIS);
long slot = sc.getSlotList()[0];
long sessionId = sc.openSession(slot);
sc.login(sessionID, PIN);

ByteArrayInputStream EncryptedInputStream = new
ByteArrayInputStream(encryptedOutputStream.toByteArray());
ByteArrayOutputStream decryptedOutputStream = new ByteArrayOutputStream();

CmsEnvelopeStreamParser cmsParser = new
CmsEnvelopeStreamParser(EncryptedInputStream);
IDecryptorStore decryptor = new SCDecryptor(sc, sessionId);
cmsParser.open(decryptedOutputStream, decryptor);
```

C#

```
SmartCard sc = new SmartCard(CardType.AKIS);
long slot = sc.getSlotList()[0];
long sessionId = sc.openSession(slot);
sc.login(sessionID, PIN);

MemoryStream EncryptedInputStream = new
MemoryStream(encryptedOutputStream.ToArray());
MemoryStream decryptedOutputStream = new MemoryStream();

CmsEnvelopeStreamParser cmsParser = new
CmsEnvelopeStreamParser(EncryptedInputStream);
IDecryptorStore decryptor = new SCDecryptor(sc, sessionId);
cmsParser.open(decryptedOutputStream, decryptor);
```

4. Çözücüler

Kullanılacak çözücülerin IDecryptorStore arayüzünden türemiş olmaları gerekmektedir. IDecryptorStore arayüzünün iki metodu bulunmaktadır. *getEncryptionCertificates()* metodu çözücünün çözebileceği sertifikaları verir. *decrypt(...)* metoduna, hangi sertifika ile şifre çözme yapılması isteniyorsa o sertifika ve çözülmek istenen veri parametre olarak verilir; metot çözülmüş veriyi geri döner. Bu arayüze uyacak şekilde kendi çözücünüzü de yazabilirsiniz.

CMS Zarf kütüphanesinde hali hazırda akıllı kartta, bellekte ve Microsoft sertifika deposu üzerinden çözme işlemleri yapan çözücüler bulunmaktadır.

4.1 Akıllı Kart Çözücü

Akıllı kart ile çözme işlemleri için SCDecryptor sınıfı kullanılmalıdır. Bu sınıf, akıllı kart işlemleri yapacağı SmartCard nesnesini ve oturum numarasını parametre olarak alır. Verinin akıllı kart ile çözülebilmesi için akıllı karta giriş (login) yapılması gerekmektedir.

Akıllı kart işlemleri hakkında daha geniş bilgi almak için MA3 API Kullanım Kılavuzunun "8. Akıllı Kart" bölümünü inceleyiniz. Aşağıda örnek bir çözücünün yaratılışı vardır.

Java

```
SmartCard sc = new SmartCard(CardType.AKIS);
long slot = sc.getSlotList()[0];
long sessionId = sc.openSession(slot);
sc.login(sessionId, PIN);
IDecryptorStore decryptor = new SCDecryptor(sc, sessionId);
```

C#

```
SmartCard sc = new SmartCard(CardType.AKIS);
long slot = sc.getSlotList()[0];
long sessionId = sc.openSession(slot);
sc.login(sessionId, PIN);
IDecryptorStore decryptor = new SCDecryptor(sc, sessionId);
```

4.2 Bellekte Çözücü

Eğer sertifika ve özel anahtara ulaşılabiliyorsa, şifrelenmiş dosyalar bellekte de çözülebilir. Bu işlem için MemoryDecryptor sınıfı kullanılmalıdır. Nesne yaratılırken sertifika ve özel anahtar çiftleri parametre olarak geçilir.

Java

```
Pair<ECertificate, PrivateKey> recipient =
    new Pair<ECertificate, PrivateKey>(cert, privKey);
IDecryptorStore decryptor = new MemoryDecryptor(recipient);
```

C#

```
Pair<ECertificate, IPrivateKey> recipient =
    new Pair<ECertificate, IPrivateKey>(cert, privKey);
IDecryptorStore decryptor = new MemoryDecryptor(recipient);
```

4.3 Microsoft Sertifika Deposundan Çözücü

Microsoft sertifika deposundaki anahtar kullanılarak şifreli veri çözülür. MSCerStoreDecryptor sınıfı kullanılarak çözme işlemi yapılabilir. Bu sınıf herhangi bir parametreye ihtiyaç duymaz. Yalnız kullanıcının sertifika deposuna erişim hakkı olması gerekmektedir.

5. Şifrelenmiş Veriye Yeni Alıcı Ekleme

Şifrelenmiş veriye yeni alıcılar eklenebilir. Bunun için şifreli dokümanın hepsini çözmeye gerek yoktur. Sadece dokümanın şifrelendiği simetrik anahtarı elde edip, yeni alıcı için bu simetrik anahtarı şifreleyip dokümana eklemek gerekmektedir. Simetrik anahtarı elde edebilmek için kütüphaneye bir çözücü verilmelidir. Simetrik anahtarı elde etme işlemi dokümanın çözülmesi gibi uzun sürmez.

Java

```
SmartCard sc = new SmartCard(CardType.AKIS);
long slot = sc.getSlotList()[0];
long session = sc.openSession(slot);
sc.login(session, PIN);

List<byte[]> certs = sc.getEncryptionCertificates(session);
ECertificate cert = new ECertificate(certs.get(0));

//Add first recipient
InputStream plainInputStream = new
ByteArrayInputStream(TestData.plainString.getBytes());
ByteArrayOutputStream envelopeWithOneRecipient = new ByteArrayOutputStream();

EnvelopeConfig config = new EnvelopeConfig ();
config.setPolicy(TestConstants.getPolicyFile());

CmsEnvelopeStreamGenerator cmsGenerator = new
CmsEnvelopeStreamGenerator(plainInputStream );
cmsGenerator.addRecipients(config, cert);
cmsGenerator.generate(envelopeWithOneRecipient);
//CMS Envelope with one recipient is generated.

//Add a new recipient to envelope
ByteArrayInputStream bais = new
ByteArrayInputStream(envelopeWithOneRecipient.toByteArray());
ByteArrayOutputStream envelopeWithTwoRecipient = new ByteArrayOutputStream();

CmsEnvelopeStreamParser cmsParser = new CmsEnvelopeStreamParser(bais);
IDecryptorStore decryptor = new SCDecryptor(sc, session);
cmsParser.addRecipientInfo(envelopeWithTwoRecipient, decryptor, cert);
```

```

sc.logout(session);

sc.closeSession(sessionID);

```

C#

```

SmartCard sc = new SmartCard(CardType.AKIS);
long slot = sc.getSlotList()[0];
long sessionID = sc.openSession(slot);
sc.login(sessionID, PIN);

List<byte[]> certs = sc.getEncryptionCertificates(sessionID);
ECertificate cert = new ECertificate(certs[0]);

using (Stream plainInputStream = new
MemoryStream(Encoding.ASCII.GetBytes(TestData.plainString)),
envelopeWithOneRecipient = new MemoryStream())
{
    EnvelopeConfig config = new EnvelopeConfig();
    config.setPolicy(TestConstants.GetPolicyFile());

    CmsEnvelopeStreamGenerator cmsGenerator = new
CmsEnvelopeStreamGenerator(plainInputStream);
    cmsGenerator.addRecipients(config, cert);
    cmsGenerator.generate(envelopeWithOneRecipient);

    //CMS Envelope with one recipient is generated.

    using (MemoryStream bais = new
MemoryStream(((MemoryStream)envelopeWithOneRecipient).ToArray()),
envelopeWithTwoRecipient = new MemoryStream())
    {
        //Add a new recipient to envelope
        CmsEnvelopeStreamParser cmsParser = new
CmsEnvelopeStreamParser(bais);
        IDecryptorStore decryptor = new SCDecryptor(sc, sessionID);
        cmsParser.addRecipientInfo(envelopeWithTwoRecipient, decryptor,
cert);
        sc.logout(sessionID);
        sc.closeSession(sessionID);
    }
}

```


6. Dizin Sisteminden Sertifika Bulma

Şifreli veri oluşturmak için şifreleme yapılacak kişinin şifreleme sertifikasının elde edilmesi gerekmektedir. Bunun için kullanılabilecek yöntemlerden bir tanesi dizin sisteminden e-posta adresiyle kullanıcı sertifikasının bulunmasıdır.

Dizin sistemi işlemleri, MA3 API kütüphanesinde MA3 API Infra projesinde bulunmaktadır. Bu proje için "ma3api-infra-....jar"ı edinmeniz gerekmektedir.

Cmsenvelope\example dizinindeki *LDAPUtil* sınıfı, bizim test sistemimizde e-posta adresinden sertifikayı bulan kod örneğini içermektedir. Kendi sisteminize göre uyarlamanız gerekmektedir.

7. Sertifika Doğrulama

Şifreli veri göndereceğimiz kişinin sertifikası, kaybolma veya çalınma gibi durumlar yüzünden iptal edilmiş olabilir. Sertifikanın iptal edilmesi durumunda, o sertifika için şifreleme yapılmaması gerekmektedir. Şifrelenmiş bir verinin şifresini çözerken ise sertifika doğrulama yapılmasına gerek yoktur.

Sertifikanın geçerliliğini kontrol eden modül "MA3 API CertValidation" modülüdür. Sertifika geçerlilik kontrolü için; Java teknolojisi kullanılıyorsa "ma3api-certvalidation.jar" .NET teknolojisi kullanılıyorsa "ma3api-certvalidation.dll" dosyasına sahip olmanız gerekmektedir. Sertifika doğrulama hakkında daha geniş bilgi almak için MA3 API Kullanım Kılavuzunun "3. Sertifika Doğrulama" bölümünü inceleyiniz.

CMS Zarf Kütüphanesi şifreleme işlemini yapmadan önce, şifrelemenin yapılacağı sertifikayı doğrular. Sertifika doğrulamanın ana unsurlarından bir tanesi de doğrulama sırasında kullanılacak öğeleri tanımlayan politika dosyasıdır. Örnek bir politika dosyasını API ile birlikte edinebilirsiniz. Politika dosyasını ihtiyaçlarınıza göre şekillendirmek için MA3 API kullanım kılavuzundaki "3.7 Politika Dosyası" bölümünü inceleyebilirsiniz. Politika dosyasını, CMS Zarf kütüphanesindeki *EnvelopeConfig* sınıfını kullanarak verebilirsiniz.

8. Donanım Tabanlı Rassal Sayı Üretme

API içerisindeki rassal sayı üreteçleri, tohum(seed) dediğimiz bir rassal veri kaynağını kullanarak rassal veri üretirler. Tohum verisinin rassallığı, nihai üretilmiş verinin rassallığını belirler.

Rastgele üretilmiş veri, CMS Zarf API'sinde simetrik şifreleme anahtarı olarak kullanıldığı için kritik öneme sahiptir. Bundan dolayı CMS Zarf API'si kullanılırken donanım tabanlı rassal sayı üretme kaynağı kullanılması önerilmektedir. Donanım olarak, rassal sayı üreticisinin güvenli olduğuna dair sertifikası olan akıllı kartlar veya HSM'ler kullanılabilir.

API'ye eklediğiniz rassal sayı kaynağı, uygulama sonlanıncaya kadar geçerli olacaktır. Dolayısıyla kaynak ekleme işleminin, uygulama yaşam döngüsünün başında ve sadece bir kere yapılması gerekmektedir. Eğer eklediğiniz kaynağın artık kullanılmamasını istiyorsanız, *Crypto.getRandomGenerator().removeAllSeeders()* fonksiyonu ile bütün kaynakları silip sadece kullanılmasını istediğiniz kaynakları eklemeniz gerekmektedir.

8.1 Akıllı Kart Tabanlı Rassal Sayı Üretme

Akıllı karttan alınan verinin, rassal sayı tohumu olarak kullanılması için *SmartCardSeed* sınıfından yararlanılabilir.

Java

```
SmartCard sc = new SmartCard(CardType.AKIS);  
long slot = sc.getSlotList()[0];  
SmartCardSeed scSeed = new SmartCardSeed(CardType.AKIS, slot);  
Crypto.getRandomGenerator().addSeeder(scSeed);
```

C#

```
SmartCard sc = new SmartCard(CardType.AKIS);  
long slot = sc.getSlotList()[0];  
SmartCardSeed scSeed = new SmartCardSeed(CardType.AKIS, slot);  
Crypto.getRandomGenerator().addSeeder(scSeed);
```

EK 1. Şifreleme API Güvenli Kullanım Tavsiyeleri

K.1 *Veri şifreleme ve çözme işlemleri, kullanım kılavuzunun 2. ve 3. bölümlerinde yer alan örneklerle uygun şekilde yapılmalıdır.*

K.2 *Donanım tabanlı, onay almış RNG kaynakları kullanılmalıdır. Detaylı bilgi ve örnek kod için kullanım kılavuzunun 8. Donanım Tabanlı Rassal Sayı Üretme bölümüne bakınız.*

K.3 *Simetrik şifreleme işlemleri için AES 256 algoritması kullanılmalıdır.*

K.4 *AES 256 simetrik şifreleme algoritması CBC modunda çalıştırılmalıdır.*

K.5 *Anahtar yönetim şekli anahtar taşıma olduğunda desteklenen algoritmalar: RSA_PKCS1, RSA_OAEP_SHA1, RSA_OAEP_SHA256 ve RSA_OAEP_SHA512 algoritmalarıdır. Güvenlik açısından minimum RSA_OAEP_SHA256 algoritması kullanılmalıdır.*

K.6 *Kullanılan asimetrik algoritmanın anahtar boyu alıcı sertifikasında tanımlıdır. RSA şifrelemeleri için minimum 2048 bit anahtar kullanılmalıdır.*

K.7 *Şifreli veriyi çözebilmek için kullanılacak özel anahtar(lar) akıllı kart veya HSM de tutulmalıdır.*

K.8 *Varsayılan politika dosyası kullandığında sertifikanın güvenilir bir kök tarafından verildiğinden ve geçerli olduğundan emin olunduğu için varsayılan politika dosyası değiştirilmemelidir.*

K.9 *Sertifika doğrulama sırasında sertifika iptal kontrolü yapılmalıdır. API ile verilen örnek politika dosyasının kullanılması bunu sağlayacaktır.*