TASNİF DIŞI

TÜBİTAK BİLGEM
KAMU SERTİFİKASYON MERKEZİ

İMZA ARŞİVLEME REHBERİ

Doküman Kodu
REH.05.04

Revizyon No
06

Revizyon Tarihi
16.09.2022

TASNİF DIŞI

REVİZYON GEÇMİŞİ		
Revizyon No	**Revizyon Nedeni**	**Revizyon Tarihi**
00	İlk çıkış	12.11.2015
01	XAdES sözde kod eklendi	01.12.2015
02	Açıklama eklendi	28.01.2016
03	Giriş sayfası yenilendi	09.02.2016
04	PAdES sözde kod eklendi	02.03.2016
05	Doküman biçimi yeni şablona aktarılmıştır. Doküman kodu güncellenmiştir. Dokümanın eski revizyonları Kamu SM doküman yönetim sisteminde "REH-001-013" kodu ile yer almaktadır.	01.06.2018
06	EYP uygulamalarında arşiv işlemi için sözde kod eklenmiştir.	16.09.2022

İÇİNDEKİLER

1 Amaç ve Kapsam

Bu doküman, imzalı dosyaların arşiv tipine dönüştürülmesi sırasında kullanılmak üzere oluşturulmuştur.

ETSI TS 101 733 CAdES, ETSI TS 101 903 XAdES ve ETSI TS 103 172 PAdES imzalı dosyaların arşivlenmesi için izlenecek yollar ve kullanılması tavsiye edilen mimari doküman kapsamında belirtilmiştir.

Bu doküman;

ETSI TS 101 733: Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)

ETSI TS 101 903: XML Advanced Electronic Signatures (XAdES)

ETSI TS 103 172: Electronic Signatures and Infrastructures (ESI); PAdES Baseline Profile standartları referans alınarak hazırlanmıştır.

2 Kısaltmalar

CAdES:	CMS Advanced Electronic Signature - CMS Gelişmiş Elektronik İmza
CMS:	Cryptographic Message Syntax- Kriptografik Mesaj Sözdizimi
ETSI:	European Telecommunications Standards Institute-Avrupa Telekomünikasyon Standartları Enstitüsü
EYP	Elektronik Yazışma Paketi
Kamu SM:	Kamu Sertifikasyon Merkezi
PAdES:	PDF Advanced Electronic Signature - PDF Gelişmiş Elektronik İmza
XAdES:	XML Advanced Electronic Signature - XML Gelişmiş Elektronik İmza
XML:	Extensible Markup Language - Genişletilebilir İşaretleme Dili
Zaman Damgası:	E-imza mevzuatında tanımlanan Zaman Damgası

3 Arşivleme Talimatı

Arşiv imza, e-imzalı belgelerin sertifika makamına ait kök/alt kök, OCSP ve zaman damgası sertifikalarının geçerlilik süresinden daha uzun bir süre saklanması gerektiği durumlarda kullanılması gereken imza tipidir.

Arşivleme, sertifika makamına ait sertifikaların geçerlilik süresinin sonuna yaklaşılması, sertifikaların iptal olması veya kullanılan algoritmaların geçerliliğini yitirmesi durumlarında yapılır. Arşivlemenin yukarıdaki durumlar oluşmadan önce yapılmasında da bir sakınca yoktur. Arşivleme, hali hazırda arşiv tipindeki imzalı dosyaların içindeki son arşiv zaman damgasının geçerliliği tehlikeye girdiği takdirde, ESHS tarafından yeni bir hiyerarşiden yayınlanmış zaman damgası ayarları girilerek tekrarlanmalıdır. Uygulama tarafında ise ilgili altyapı sağlanmış olmalıdır.

Aşağıdaki bölümde arşivleme ihtiyacı gerektirecek senaryolar belirlenmiştir.

Arşivleme Senaryoları:

1. Sertifika ile ilgili senaryolar
    a. OCSP sertifikasının iptal olma ve süresinin dolma durumu
    b. İmza ZD sertifikasının iptal olma ve süresinin dolma durumu
    c. “İmza ve referans zaman damgası” ya da “referans zaman damgası” sertifikasının (eğer imzada varsa) iptal olma ve süresinin dolma durumu
    d. Arşiv ZD sertifikasının süresinin dolma durumu
    e. Alt kök sertifikasının iptal olma ve süresinin dolma durumu
    f. Kök sertifikasının süresinin dolma durumu
2. Güvenilir kök deposu değişiklikleri ile ilgili senaryolar
    a. Kök Sertifikasının kara listeye girme durumu
3. Algoritma geçersizlikleri ile ilgili senaryolar
    a. İmza zaman damgası algoritmasının geçersiz olma durumu
    b. “İmza ve referans zaman damgası” ya da “referans zaman damgası” (eğer imzada varsa) algoritmasının geçersiz olma durumu
    c. Arşiv zaman damgası algoritmasının geçersiz olma durumu
    d. İmza algoritmalarının geçersiz olma durumu

Senaryoları sağlamak adına kurulması istenen işleyiş aşağıda anlatılmakta ve sözde (pseudo) kodu verilmektedir.

1. Sistemin arka planında çalışacak uygulama kullanıcıdan bağımsız olarak toplu işlem (batch process) yapmalıdır.
2. İçerisine güvenilirliğini yitirmiş kök sertifikaların özet değerlerinin eklenebileceği kara liste (blacklist) yapısı kurulmalıdır.
3. Güvenli algoritmaların eklenebileceği beyaz liste (whitelist) yapısı kurulmalıdır.

Arşivleme testlerinin tarafımızca yapılabilmesi için toplu işlemi (batch process) tetikleyip logların alınabileceği ve sonrasında arşivlenen dosyaların indirilebileceği geçici basit bir arayüz yapılmalıdır. Arayüz, kara listeye eklenen kök sertifikalarını ve beyaz listeye eklenen özet algoritmalarını görmeye imkan vermelidir.

3.1 EBYS Uygulamaları için Arşivleme Sözde Kodu

CAdES API'de arşivleme işleminin aşağıda yazan sözde koda göre gerçeklenmesi tavsiye edilmektedir.

```
//Arşiv kontrolünden geçecek imzalar toplanır ve tek tek kontrolden geçirilir.
List<Signature> signatureFileList = getAllSignatures();

foreach(Signature s in signatureFileList){
    //Öncelikle imza doğrulaması yapılır, imza doğrulanmadığı takdirde arşivlenmez ve hata loglanır.
    //İmza doğrulama işleminde algoritmalar için whitelist ve sertifikalar için blacklist kontrolü
    //yapılmaz.
    if (verifySignature(s)) {

        //İlk seviyedeki bütün paralel imzacılar alınır.
        List<Signer> parallelSigList = s.getParallelSignerList();

        //Belirlenen paralel imzacıların imzaları tek tek kontrol edilir.
        foreach (Signer parallelSig in parallelSigList) {

            //Paralel imzanın tipi alınır.
            SignatureType parallelSigType = parallelSig.getType();

            //Kontrol edilen paralel imzacının bütün alt imzacıları DFS(Depth First Search)
            //veya BFS(Breadth First Search) ile alınır.
            //Paralel imzacının kendisi listeye dahil değildir.
            List<Signer> subSignerList = getAllSubSigners(parallelSig);

            //Listede tipi XLONG olmayan imza varsa önce paralel imzacı tipi Arşivse
            //XLONG tipine çevrilir.
            //Sonra paralel imzacıya ait alt imzalardan XLONG tipinde olmayan imzalar
            //XLONG tipine çevrilir.
            foreach (Signer subSigner in subSignerList) {
                if (subSigner.getType() != SignatureType.ESXLong) {

                    //Paralel imza tipi Arşivse XLONG tipine çevrilir.
                    if (parallelSigType == SignatureType.ESA) {
                        downgradeToXLong(parallelSig);
                        parallelSigType = SignatureType.ESXLong;
                        log("Arşiv imza XLONG tipine çevrildi.");
                    }

                    //Alt imza XLONG tipine çevrilir.
                    upgradeToXLong(subSigner);
                    log("Alt imza XLONG tipine çevrildi.");
                }
            }

            //Paralel imza arşiv tipinde değilse arşivlenir.
            if (parallelSigType != SignatureType.ESA) {
                archive(parallelSig);
                log("Paralel imza arşiv tipinde olmadığı için arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgasının algoritmaları alınır.
            List<String> lastArchiveTSAlgorithmList = getLastArchiveTSAlgorithms(parallelSig);

            //Geçerli algoritma listesi alınır. Eğer yakın zamanda geçersiz olacak algoritma varsa
            //bu listeden çıkarılmalıdır.
            //Son arşiv zaman damgasındaki algoritmalar geçerli algoritma listesiyle karşılaştırılır.
            boolean isAlgorithmInWhiteList = isAlgorithmInWhiteList(lastArchiveTSAlgorithmList);

            //Son arşiv zaman damgasında geçersiz algoritma varsa imza arşivlenir.
            if (!isAlgorithmInWhiteList) {
                archive(parallelSig);
                log("Son arşiv zaman damgasında geçersiz algoritma bulunması sebebiyle imza yeniden
arşivlendi.");
                continue;
            }
```

```
            //Son arşiv zaman damgası sertifikası alınır.
            Certificate lastArchiveTSCertificate = getLastArchiveTSCertificate(parallelSig);

            //Son arşiv zaman damgası sertifika süresinin dolmasına 2 aydan az kaldıysa yeni arşiv
           //zaman damgası ayarları yapılır ve imza arşivlenir.
            Date certificateExpirationDate = getCertificateExpirationDate(lastArchiveTSCertificate);
            if (certificateExpirationDate < Date.now + 2 months) {
                newArchiveTsSettings();
                archive(parallelSig);
                log("Sertifika süresinin dolmasına 2 aydan az kaldığı için imza yeniden
arşivlendi.");
                continue;
            }
            //Son arşiv zaman damgası sertifikası doğrulanamazsa yeni arşiv zaman damgası ayarları
            //yapılır ve imza arşivlenir.
            if (!verifyCertificate(lastArchiveTSCertificate)) {
                newArchiveTsSettings();
                archive(parallelSig);
                log("Sertifika doğrulanamadığı için imza yeniden arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgası kök sertifikası alınır.
            Certificate lastArchiveTSRootCertificate =
getLastArchiveTSRootCertificate(lastArchiveTSCertificate);
            //Son arşiv zaman damgası kök sertifikasının kara listede olup olmadığına bakılır.
            //İptal olan kök sertifikalar öncelikle bu listeye eklenmiş olmalıdır.
            boolean isInRootCertificateBlackList =
isInRootCertificateBlackList(lastArchiveTSRootCertificate);

            //Son arşiv zaman damgası kök sertifikası kara listedeyse yeni arşiv zaman damgası
            //ayarları yapılır ve imza arşivlenir.
            if (isInRootCertificateBlackList) {
                newArchiveTsSettings();
                archive(parallelSig);
                log("Son arşiv zaman damgası kök sertifikası kara listede olduğu için imza yeniden
arşivlendi.");
                continue;
            }

            log("İmzanın arşivlenmesine gerek yok.");
        }
    } else
        log("İmza doğrulanamadığı için arşivlenmedi.");
}
```

XAdES API'de arşivleme işleminin aşağıda yazan sözde koda göre gerçeklenmesi tavsiye edilmektedir.

```
//Arşiv kontrolünden geçecek imzalı dosyalar toplanır ve tek tek kontrolden geçirilir.

List<Signature> signatureFileList = getAllSignatures();

foreach(Signature s in signatureFileList){
    //Öncelikle imza doğrulaması yapılır, imza doğrulanmadığı takdirde arşivlenmez ve hata loglanır.
    //İmza doğrulama işleminde algoritmalar için whitelist ve sertifikalar için blacklist kontrolü
    //yapılmaz.
    if (verifySignature(s)) {
        //İlk seviyedeki bütün paralel imzalar alınır.
        List<Signature> parallelSigList = s.getParallelSignatureList();

        //Belirlenen paralel imzalar tek tek kontrol edilir.
        foreach (Signature parallelSig in parallelSigList) {

            //Paralel imzanın tipi alınır.
            SignatureType parallelSigType = parallelSig.getType();

            //Kontrol edilen paralel imzanın bütün alt imzaları DFS(Depth First Search)
            //veya BFS(Breadth First Search) ile alınır.
            //Paralel imzanın kendisi listeye dahil değildir.
            List<Signature> subSignatureList = getAllSubSignatures(parallelSig);

            //Listede tipi XLONG olmayan imza varsa önce paralel imzanın tipi Arşivse XLONG tipine
            //çevrilir.
            //Sonra paralel imzaya ait alt imzalardan XLONG tipinde olmayan imzalar XLONG tipine
            //çevrilir.
            foreach (Signature subSignature in subSignatureList) {
                if (subSignature.getType() != SignatureType.ESXLong) {

                    //Paralel imza tipi Arşivse XLONG tipine çevrilir.
                    if (parallelSigType == SignatureType.ESA) {
                        downgradeToXLong(parallelSig);
                        parallelSigType = SignatureType.ESXLong;
                        log("Arşiv imza XLONG tipine çevrildi.");
                    }
                    //Alt imza XLONG tipine çevrilir.
                    upgradeToXLong(subSignature);
                    log("Alt imza XLONG tipine çevrildi.");
                }
            }

            //Paralel imza arşiv tipinde değilse arşivlenir.
            if (parallelSigType != SignatureType.ESA) {
                archive(parallelSig);
                log("Paralel imza arşiv tipinde olmadığı için arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgasının algoritmaları alınır.
            List<String> lastArchiveTSAlgorithmList = getLastArchiveTSAlgorithms(parallelSig);
            //Geçerli algoritma listesi alınır. Eğer yakın zamanda geçersiz olacak algoritma varsa
            //bu listeden çıkarılmalıdır.
            //Son arşiv zaman damgasındaki algoritmalar geçerli algoritma listesiyle karşılaştırılır.
            boolean isAlgorithmInWhiteList = isAlgorithmInWhiteList(lastArchiveTSAlgorithmList);

            //Son arşiv zaman damgasında geçersiz algoritma varsa imza arşivlenir.
            if (!isAlgorithmInWhiteList) {
                archive(parallelSig);
                log("Son arşiv zaman damgasında geçersiz algoritma bulunması sebebiyle imza yeniden
arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgası sertifikası alınır.
            Certificate lastArchiveTSCertificate = getLastArchiveTSCertificate(parallelSig);
```

```
//Son arşiv zaman damgası sertifika süresinin dolmasına 2 aydan az kaldıysa yeni arşiv zaman damgası
//ayarları yapılır ve imza arşivlenir.
            Date certificateExpirationDate = getCertificateExpirationDate(lastArchiveTSCertificate);
            if (certificateExpirationDate < Date.now + 2 months) {
                newArchiveTsSettings();
                archive(parallelSig);
                log("Sertifika süresinin dolmasına 2 aydan az kaldığı için imza yeniden
arşivlendi.");
                continue;
            }
            //Son arşiv zaman damgası sertifikası doğrulanamazsa yeni arşiv zaman damgası ayarları
            //yapılır ve imza arşivlenir.
            if (!verifyCertificate(lastArchiveTSCertificate)) {
                newArchiveTsSettings();
                archive(parallelSig);
                log("Sertifika doğrulanamadığı için imza yeniden arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgası kök sertifikası alınır.
            Certificate lastArchiveTSRootCertificate =
getLastArchiveTSRootCertificate(lastArchiveTSCertificate);
            //Son arşiv zaman damgası kök sertifikasının kara listede olup olmadığına bakılır.
            //İptal olan kök sertifikalar öncelikle bu listeye eklenmiş olmalıdır.
            boolean isInRootCertificateBlackList =
isInRootCertificateBlackList(lastArchiveTSRootCertificate);

            //Son arşiv zaman damgası kök sertifikası kara listedeyse yeni arşiv zaman damgası
            //ayarları yapılır ve imza arşivlenir.
            if (isInRootCertificateBlackList) {
                newArchiveTsSettings();
                archive(parallelSig);
                log("Son arşiv zaman damgası kök sertifikası kara listede olduğu için imza yeniden
arşivlendi.");
                continue;
            }

            log("İmzanın arşivlenmesine gerek yok.");
        }
    } else
        log("İmza doğrulanamadığı için arşivlenmedi.");
}
```

PAdES API'de arşivleme işleminin aşağıda yazan sözde koda göre gerçeklenmesi tavsiye edilmektedir.

```
//Arşiv kontrolünden geçecek imzalı dosyalar toplanır ve tek tek kontrolden geçirilir.
List<Signature> signatureFileList = getAllSignatures();

foreach(Signature s in signatureFileList) {
    //Öncelikle imza doğrulaması yapılır, imza doğrulanmadığı takdirde arşivlenmez ve hata loglanır.
    if (verifySignature(s)) {
        DSS dssToBeAdded=new DSS();
        SignatureType subSignatureType;
        Signature lastSignature;
        boolean needDocumentTS=false;

        //Kontrol edilen imzanın bütün alt imzaları alınır.
        //İmzalar imza sırasına göre listeye eklenmelidir.
        List<Signature> subSignatureList = getAllSubSignatures(s);

        //Listede bulunan imzalarda doğrulama verileri var olmayan imza varsa
        //doğrulama verileri yeni tanımlanan DSS içerisine eklenir.
        foreach (Signature subSignature in subSignatureList) {
            subSignatureType=subSignature.getSignatureType();
                if(subSignatureType != SignatureType.LT && subSignatureType != SignatureType.LTA){
                    if(subSignatureType==SignatureType.B_Type)
                        needDocumentTS=true;

                    addValidationData(subSignature, dssToBeAdded);
                }

            lastSignature=subSignature;
        }

        if(needDocumentTS) {
            addDocumentTS(lastSignature)
        }

        if(dssToBeAdded!=null) {
            //dssToBeAdded içerisinde dublicate olan veriler temizlenir.
            removeDublicate(dssToBeAdded);
            addDSSToSignature(dss,lastSignature);
            archive(lastSignature);
            log("İmza arşiv tipinde olmadığı için arşivlendi.");
            continue;
        }
        else {
            //Son arşiv zaman damgasının algoritmaları alınır.
            String lastArchiveTSAlgorithmList = getLastArchiveTSAlgorithms(lastSignature);

            //Geçerli algoritma listesi alınır. Eğer yakın zamanda geçersiz olacak algoritma varsa
            //bu listeden çıkarılmalıdır.
            //Son arşiv zaman damgasındaki algoritmalar geçerli algoritma listesiyle karşılaştırılır.
            boolean isAlgorithmInWhiteList = isAlgorithmInWhiteList(lastArchiveTSAlgorithmList);

            //Son arşiv zaman damgasında geçersiz algoritma varsa imza arşivlenir.
            if (!isAlgorithmInWhiteList) {
                newArchiveTsSettings();
                archive(lastSignature);
                log("Son arşiv zaman damgasında geçersiz algoritma bulunması sebebiyle imza yeniden
arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgası sertifikası alınır.
            Certificate lastArchiveTSCertificate = getLastArchiveTSCertificate(s);

            //Son arşiv zaman damgası sertifika süresinin dolmasına 2 aydan az kaldıysa yeni arşiv
            //zaman damgası ayarları yapılır ve imza arşivlenir.
            Date certificateExpirationDate = getCertificateExpirationDate(lastArchiveTSCertificate);
            if (certificateExpirationDate < Date.now + 2 months) {
                newArchiveTsSettings();
                archive(lastSignature);
```

```
                log("Son arşiv zaman damgası sertifikasının süresinin dolmasına 2 aydan az kaldığı
için imza yeniden arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgası sertifikası doğrulanamazsa yeni arşiv zaman damgası ayarları
            //yapılır ve imza arşivlenir.
            if (!verifyCertificate(lastArchiveTSCertificate)) {
                newArchiveTsSettings();
                archive(lastSignature);
                log("Son arşiv zaman damgası sertifikası doğrulanamadığı için imza yeniden
arşivlendi.");
                continue;
            }

            //Son arşiv zaman damgası kök sertifikası alınır.
            Certificate lastArchiveTSRootCertificate =
getLastArchiveTSRootCertificate(lastArchiveTSCertificate);

            //Son arşiv zaman damgası kök sertifikasının kara listede olup olmadığına bakılır.
            //İptal olan kök sertifikalar öncelikle bu listeye eklenmiş olmalıdır.
            boolean isInRootCertificateBlackList =
isInRootCertificateBlackList(lastArchiveTSRootCertificate);

            //Son arşiv zaman damgası kök sertifikası kara listedeyse yeni arşiv zaman damgası
            //ayarları yapılır ve imza arşivlenir.
            if (isInRootCertificateBlackList) {
                newArchiveTsSettings();
                archive(lastSignature);
                log("Son arşiv zaman damgası kök sertifikası kara listede olduğu için imza yeniden
arşivlendi.");
                continue;
            }

            log("İmzanın arşivlenmesine gerek yok.");
        }
    }
    else
        log("İmza doğrulanamadığı için arşivlenmedi.");
}
```

3.2 EYP Uygulamaları için Arşivleme Sözde Kodu

EYP için CAdES API'de arşivleme işleminin aşağıda yazan sözde koda göre gerçeklenmesi tavsiye edilmektedir.

```
//Arşiv kontrolünden geçecek mühürler toplanır ve tek tek kontrolden geçirilir.
List<Signature> signatureFileList = getAllSignatures();

foreach(Signature s in signatureFileList){
    //Öncelikle imza doğrulaması yapılır, imza doğrulanmadığı takdirde arşivlenmez ve hata loglanır.
    //İmza doğrulama işleminde algoritmalar için whitelist ve sertifikalar için
    //blacklist kontrolü yapılmaz.
    if (verifySignature(s)) {

        //Son arşiv zaman damgasının algoritmaları alınır.
        List<String> lastArchiveTSAlgorithmList = getLastArchiveTSAlgorithms(s);

        //Geçerli algoritma listesi alınır. Eğer yakın zamanda geçersiz olacak algoritma varsa
        //bu listeden çıkarılmalıdır.
        //Son arşiv zaman damgasındaki algoritmalar geçerli algoritma listesiyle karşılaştırılır.
        boolean isAlgorithmInWhiteList = isAlgorithmInWhiteList(lastArchiveTSAlgorithmList);

        //Son arşiv zaman damgasında geçersiz algoritma varsa imza arşivlenir.
        if (!isAlgorithmInWhiteList) {
            archive(s);
            log("Son arşiv zaman damgasında geçersiz algoritma bulunması sebebiyle imza yeniden
arşivlendi.");
            continue;
        }

        //Son arşiv zaman damgası sertifikası alınır.
        Certificate lastArchiveTSCertificate = getLastArchiveTSCertificate(s);

        //Son arşiv zaman damgası sertifika süresinin dolmasına 2 aydan az kaldıysa yeni arşiv zaman
        //damgası ayarları yapılır ve imza arşivlenir.
        Date certificateExpirationDate = getCertificateExpirationDate(lastArchiveTSCertificate);
        if (certificateExpirationDate < Date.now + 2 months) {
            newArchiveTsSettings();
            archive(s);
            log("Sertifika süresinin dolmasına 2 aydan az kaldığı için imza yeniden arşivlendi.");
            continue;
        }
        //Son arşiv zaman damgası sertifikası doğrulanamazsa yeni arşiv zaman damgası ayarları
        //yapılır ve imza arşivlenir.
        if (!verifyCertificate(lastArchiveTSCertificate)) {
            newArchiveTsSettings();
            archive(s);
            log("Sertifika doğrulanamadığı için imza yeniden arşivlendi.");
            continue;
        }

        //Son arşiv zaman damgası kök sertifikası alınır.
        Certificate lastArchiveTSRootCertificate =
getLastArchiveTSRootCertificate(lastArchiveTSCertificate);
        //Son arşiv zaman damgası kök sertifikasının kara listede olup olmadığına bakılır. İptal olan
        //kök sertifikalar öncelikle bu listeye eklenmiş olmalıdır.
        boolean isInRootCertificateBlackList =
isInRootCertificateBlackList(lastArchiveTSRootCertificate);

        //Son arşiv zaman damgası kök sertifikası kara listedeyse yeni arşiv zaman damgası ayarları
        //yapılır ve imza arşivlenir.
        if (isInRootCertificateBlackList) {
            newArchiveTsSettings();
            archive(s);
            log("Son arşiv zaman damgası kök sertifikası kara listede olduğu için imza yeniden
arşivlendi.");
            continue;
        }
        log("İmzanın arşivlenmesine gerek yok.");
    } else
        log("İmza doğrulanamadığı için arşivlenmedi.");
}
```