TASNİF DIŞI

TÜBİTAK BİLGEM
KAMU SERTİFİKASYON MERKEZİ

EBYS UYUM DEĞERLENDİRME ÖN HAZIRLIK REHBERİ

Doküman Kodu
REH.05.01

Revizyon No
06

Revizyon Tarihi
01.06.2018

TASNİF DIŞI

REVİZYON GEÇMİŞİ		
Revizyon No	**Revizyon Nedeni**	**Revizyon Tarihi**
00	İlk çıkış.	27.04.2015
01	Üst bilgi ve içerik düzenlendi.	29.01.2016
02	Uyum Değerlendirme Test Süreci hakkındaki bilgiler güncellendi.	18.03.2016
03	İmza Oluşturma Testleri Ön İsterleri ve İmza Doğrulama Testleri Ön İsterleri bölümleri güncellendi. İmza Arşivleme Rehberi eklendi.	05.06.2017
04	Arayüz ile ilgili kontroller eklendi.	15.10.2017
05	Güvenilir algoritmalar için beyaz liste kavramı eklendi. İmzanın sunucuda yükseltilmesiyle ilgili madde eklendi. Doküman kodu güncellendi.	08.12.2017
06	Doküman biçimi yeni şablona aktarılmıştır. Doküman kodu güncellenmiştir. Dokümanın eski revizyonları Kamu SM doküman yönetim sisteminde “REH-001-007” kodu ile yer almaktadır.	01.06.2018

İÇİNDEKİLER

1 Amaç ve Kapsam 3

2 Kısaltmalar 4

3 İmza Oluşturma Testleri Ön İsterleri 5

4 İmza Doğrulama Testleri Ön İsterleri 9

5 Uyum Değerlendirme Test Süreci Hakkında Bilgilendirmeler 11

6 Ek-A İmza Arşivleme Rehberi 12

1 Amaç ve Kapsam

Bu doküman, e-imza uyum değerlendirme çalışması öncesi kuruma gönderilen test paketinin içeriği hakkında bilgi vermek ve kurum tarafından ön hazırlık olarak yerine getirilmesi gereken şartları belirtmek için oluşturulmuştur. Uyum değerlendirme çalışması format kontrolü, imza oluşturma, doğrulama ve arşivleme testleri olmak üzere dört bölümden oluşmaktadır. Testlerin ayrıntılı biçimde yapılabilmesi için Kamu SM tarafından hazırlanmış olan Kamu SM Test Suit çalışması kullanılmaktadır.

Kamu kurumlarının Elektronik Belge Yönetim Sistemlerinde (EBYS) yapılacak olan e-imza uyum değerlendirme çalışması, ilgili EBYS üzerinde çalışan elektronik imza mekanizmasının uluslararası standartlara uygunluğunun kontrolünden ibarettir. İmza uygulamasının oluşturduğu imzalı dosya tiplerinin ES X-Long veya ES-A olması gerekmektedir.

Bu doküman;

CWA 14170: Security Requirements for Signature Creation Applications (İmza Oluşturma Uygulamaları için Güvenlik Gereksinimleri),

CWA 14171: Procedures for Electronic Signature Verification (Elektronik İmza Doğrulama için Prosedürler),

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

BES:	Basic Electronic Signature - Basit Elektronik İmza
CAdES:	CMS Advanced Electronic Signature - CMS Gelişmiş Elektronik İmza
CMS:	Cryptographic Message Syntax- Kriptografik Mesaj Sözdizimi
CWA:	CEN (Comité Européen De Normalisation) Workshop Agreement-CEN Çalıştay Kararları
EBYS:	Elektronik Belge Yönetim Sistemleri
ES-A:	Archival Electronic Signature - Arşiv Elektronik İmza
ES X-Long:	EXtended Long Electronic Signature - Genişletilmiş Uzun Elektronik İmza
ETSI:	European Telecommunications Standards Institute-Avrupa Telekomünikasyon Standartları Enstitüsü
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 İmza Oluşturma Testleri Ön İsterleri

1. TÜBİTAK API kullanan uygulamalar API'nin son sürümünü ve son sürümü barındıran paketten çıkan politika dosyasını kullanmalıdır. API'nin son versiyonuna https://yazilim.kamusm.gov.tr/ adresinden ulaşabilirsiniz. Eski sürüm API kullanan uygulamalar değerlendirilmeyecektir.

2. Testlerde Kamu SM Test Suit'in kullanılabilmesi için size gönderilen, RootCerts.rar klasöründe bulunan test köklerinin uygulamanın güvenli kökler dizinine eklenmesi gerekmektedir. TÜBİTAK API kullananlar ilgili politika dosyası içerisinde aşağıdaki düzenlemeyi yaparak bu hazırlığı tamamlayabilirler.

```
<trustedcertificate>
        <class
        name="tr.gov.tubitak.uekae.esya.api.certificate.validation.
        find.certificate.
        trusted.TrustedCertificateFinderFromFileSystem">
                <param name="dizin" value="D:\Trusted\" />
        </class>
        <class
        name="tr.gov.tubitak.uekae.esya.api.certificate.validation.
        find.certificate.trusted.TrustedCertificateFinderFromECertSt
        ore">
                <param name="securitylevel" value="legal" />
        </class>
</trustedcertificate>
```

Politika dosyasında ayrıca aşağıdaki ayarların yapılması gerekmektedir.

```
Bu kısım yorum satırına çekilmelidir.

<!--class name =
"tr.gov.tubitak.uekae.esya.api.certificate.validation.find.crl.CRLFi
nderFromECertStore"/>

<class name =
"tr.gov.tubitak.uekae.esya.api.certificate.validation.find.crl.CRLFi
nderFromECertStore">
<param name = "getactivecrl" value="true"/>
</class -->
```

3. Doğrudan uzun dönemli imza oluşturulduğu takdirde zaman damgası alırken sistemsel bazı aksaklıklar oluşabileceğinden imza oluşturma işlemi gerçekleşemeyebilir. Kullanıcı tarafında basit elektronik imza oluşturulduktan sonra sunucuda uzun dönemli imzaya çevrilerek bu aksaklıkların önüne geçilmesi tavsiye edilmektedir.

4. İmza oluşturma testleri, sadece imzanın formatının yukarıda belirtilen uluslararası standartlara uygunluğunun kontrolünü değil sertifika doğrulama kontrollerini de içermektedir. Bu nedenle imza uygulaması bu kontrolleri yerine getirmelidir.

5. Sertifika doğrulama, yanlış PIN girme, bloke olmuş kartla imzalama ve benzeri hatalı durumlarla ilgili kullanıcı bilgilendirmelerinde standart hata kabul edilmeyecektir. Kullanıcı bilgilendirmeleri hatayı net ifade edecek şekilde olmalıdır.

6. İmza oluşturma esnasında, birden fazla sertifikanın bulunduğu kartlarda, kullanıcıya imza oluşturma işlemi için karttaki sertifikaları seçme hakkı verilmeli ve **sadece nitelikli sertifikalar gösterilmelidir**. Sertifika seçim ekranı akıllı kartın çıkarılması, yeni kart takılması vb. durumlarda yenilenmelidir. (Bu işlem, yenileme butonu, kart okuyucunun sürekli olarak dinlenmesi gibi yöntemler ile yapılabilir.)
7. Uygulamanın, sertifika içeriğini kullanıcı istediği takdirde gösterecek şekilde geliştirilmesi gerekmektedir. Uygulama sertifikayı, işletim sisteminin sertifika görüntüleyicisi vasıtasıyla da gösterebilir.
8. İmza oluşturma işleminin yapıldığı ekrana, "**Bu imza 5070 Sayılı Elektronik İmza Kanunu'na göre güvenli elektronik imzadır."** ibaresi eklenmelidir. Kullanıcıya imzalamadan vazgeçme seçeneği sunulmalıdır.
9. Kullanıcıya imza oluşturma esnasında, imzaladığı içeriği değiştirilemez bir şekilde görüntüleme imkânı verilmelidir.
10. İmzalanmasına izin verilen belge türleri, Kalkınma Bakanlığı tarafından yayımlanan Birlikte Çalışabilirlik Esasları Rehberi'nin güncel sürümünün "2.1. Dosya Sunumu ve Değişimi" bölümünde tanımlanmaktadır. Metin Tabanlı Belgeler için PDF/A, Sayısal Grafik ve Diyagramlar için GIF, Sayısal Fotoğraflar için JPEG belge türlerine izin verilmektedir. Bunlar dışındaki belge türlerinin imzalanması uygulama tarafından kabul edilmemelidir.
11. Uygulama tarafından imzalanmasına izin verilen belge türü olarak PDF/A-1 ve PDF/A-2 belgeleri doğrudan kabul edilebilirken; PDF/A-3 belgeleri PDF/A uyumlu olmayan ek içerebildiğinden uygulamanın aşağıdaki yöntemlerden birini tercih etmesi gerekmektedir:
    a. Belge formatı PDF/A-3 olduğu için imzalanmasına izin vermemelidir.
    b. İmzalanacak belgenin PDF/A dışında ek içerdiğini tespit etmeli ve imzalanmasına izin vermemelidir.
12. Uygulamada imzacının PIN bilgisinin saklanması son derece tehlikelidir. PIN bilgisinin ve son kullanıcı bilgisayarındaki hakların saldırganlar tarafından ele geçirilmesi halinde kişi adına imza oluşturulabilir. Bu sebeple PIN bilgisinin saklanması tavsiye edilmez.
13. PIN bilgisi saklandığı takdirde imza oluşturma uygulamasının, karta "log in" olduktan belirli bir süre sonra (maksimum 30 dakika) "log out" olması gerekmektedir. Uygulamanın bir kez "log in" olup, sınırsız imzalamaya izin vermesi kabul edilmemektedir. Güvenlik açısından önerilen metod, imzalama işlemi bittikten sonra hemen "log out" olunmasıdır.
14. Sertifika seçimi yapılmadan PIN girişine izin verilmemelidir. İmza oluşturma uygulamasındaki PIN girme alanının, PIN girilmeye başladıktan bir süre sonra imzalama işlemi bitirilmediği takdirde temizlenmesi gerekmektedir. PIN alanını temizlemek için bekleme süresi maksimum 30 saniyedir.
15. Uygulama arayüzünde imzalı ve imzasız belgeler ayırt edilebilir olmalıdır. Belgenin niteliğine göre ilgili seçenekler gösterilmelidir.
16. Kullanıcı, imzalama işleminden sonra imzanın oluşup oluşmadığına dair anlaşılır bir şekilde bilgilendirilmelidir.

17. Uygulama, imza oluşturma aşamasında zaman damgası alırken, zaman damgasının geçerlilik kontrolünü yapmalıdır.

    TÜBİTAK API kullananlar bu özelliği aktive etmek için imza oluşturma kodundaki parametrelere aşağıda belirtilen eklemeleri yapmalıdır; PAdES ve Ortak API için bir değişiklik yapılmasına gerek yoktur:

CAdES	`params.put(EParameters.P_VALIDATE_TIMESTAMP_WHILE_SIGNING, true);`
XAdES	`context.setValidateTimeStamps(true);`

    TÜBİTAK API'de BES imzadan ES X-Long imzaya dönüşüm yapılıyor ise dönüştürme aşamasında bu parametre eklenmelidir.
    Testte kullanılacak zaman damgası sunucularının erişim bilgileri aşağıda verilmiştir.

***Kullanıcı Adı:** 1*	***Şifre:** 12345678 (Tüm hesaplar için kullanıcı adı ve şifre aynıdır.)*
TSA1: http://zdsA1.test2.kamusm.gov.tr	**TSB:** http://zdsB.test2.kamusm.gov.tr
TSA2: http://zdsA2.test2.kamusm.gov.tr	**TSC1:** http://zdsC1.test2.kamusm.gov.tr
TSA3: http://zdsA3.test2.kamusm.gov.tr	**TSC2:** http://zdsC2.test2.kamusm.gov.tr
TSA4: http://zdsA4.test2.kamusm.gov.tr	**TSC3:** http://zdsC3.test2.kamusm.gov.tr
TSA5: http://zdsA5.test2.kamusm.gov.tr	

    Sunucu erişim bilgilerinin kod içeriğinden ayarlanması ve uyum değerlendirme testlerinden sonra değişmesi durumunda uygulama özet değeri bozulacaktır. Bu nedenle zaman damgası sunucu erişim bilgilerinin koddan bağımsız bir şekilde konfigüre edilmesi tavsiye edilmektedir.

18. Sertifika içeriğinde maddi limit bilgisi olduğu takdirde sertifika doğrulama yapılırken aşağıdaki yöntemlerden biri izlenmelidir:
    a. Sertifikada bulunan maddi limit ile belgenin maddi içeriğinin karşılaştırılmadığı durumda, kullanıcıya imzacı sertifikasında maddi limit bilgisi olduğu uyarısı verilmeli ve imzanın oluşturulup oluşturulmayacağı EBYS uygulaması politikalarına göre belirlenmelidir.
    b. Sertifikada bulunan maddi limit ile belgenin maddi içeriğinin karşılaştırıldığı durumda:
        i. Sertifika maddi limiti belge maddi değeri için yeterli ise imza oluşturulmalıdır.
        ii. Sertifika maddi limiti belge maddi değeri için yeterli değil ise imza oluşturulmamalıdır.

19. Uygulama toplu imzalamayı destekliyorsa aşağıdaki maddeler sağlanmalıdır:
    a. Toplu imzalama sürecinde kullanıcıya özgü toplu imzalama listesi oluşturulabilmelidir. Toplu imzalama listesi imzalanacak belgelerden veya belge süreçlerinden oluşacak şekilde tasarlanmalıdır. Belge süreci, aynı belgeye ait ek, nüsha vb. bileşenlerin tamamıdır.
    b. Kullanıcı imzalanacak belge sürecini detaylı görüntüleyebilmeli ve toplu imzalama listesine eklemek isterse, detaylı görüntüleme ekranından ekleyebilmelidir.
    c. Toplu imzalama listesine eklenen belgeler kullanıcı hatasına mahal verecek şekilde toplu olarak seçilememelidir.

d. Toplu imzalama listesi görüntüleme ekranında listedeki belgelerin adı, konu başlığı ve listeye eklenme tarihi görülmelidir.
e. Uygulamada boşta bekleme süresi (maksimum 30 dakika) belirlenmelidir. Bu süre boyunca işlem yapılmadığında toplu imzalama listesi inaktif olmalıdır.
f. Uygulamada güvenli bekleme süresi maksimum 8 saat olarak belirlenmelidir. Bu süre sonunda toplu imzalama listesi sıfırlanmalıdır.

20. Uygulama arşivleme modulüne sahip olmalıdır. Arşivleme modulü Ek-A'da verilen "İmza Arşivleme Rehberi"nde belirtilen talimatlara uygun olarak geliştirilmelidir.

4 İmza Doğrulama Testleri Ön İsterleri

1. Talep edilen imza türüne göre göndermiş olduğumuz imzalı dosya paketini EBYS'nize dâhil etmeniz ve doğrulama testleri için hazır hale getirmeniz gerekmektedir.
2. Uygulama arayüzünde imzalı ve imzasız belgeler ayırt edilebilir olmalıdır. Belgenin niteliğine göre ilgili seçenekler gösterilmelidir.
3. Uygulama, doğrulanacak imzalı belgenin seçilmesine ve imzalanan içeriğin gösterilmesine imkan vermelidir.
4. İmza doğrulama ekranında imzacının isim bilgisi, imza zamanı ve imzanın genel doğrulama sonucu açıkça belirtilmelidir. Kullanıcı istediği takdirde imzacının sertifika bilgilerinin ve doğrulama sonuçlarının tamamını ayırt edilebilir şekilde görüntüleyebilmelidir. Uygulama sertifikayı işletim sisteminin sertifika görüntüleyicisi vasıtasıyla da gösterebilir.
5. Seri/Paralel imzacılar, doğrulama ekranında hiyerarşik bir düzende ağaç yapısına benzer şekilde gösterilmeli ve imzalar doğrulanırken kullanıcıyı net bir şekilde bilgilendirecek genel doğrulama sonucu ve imzacıların ayrı ayrı doğrulama sonuçları yer almalıdır.
6. İmza doğrulama sonuçları kullanıcıya anlaşılır şekilde gösterilmelidir. Kullanıcı bilgilendirmelerinde standart hata dönülmemeli, bilgilendirmeler hatayı net ifade edecek şekilde olmalıdır.
7. İmzaya dahil olan imza özelliklerinden "ContentHints" imza özelliği ile imzalanan dosya türü karşılaştırılmalıdır. Eşleşmediği durumlarda imza doğrulanmamalıdır ve kullanıcı bilgilendirilmelidir.
8. Uygulama ayrık imza oluşturuyor ise sisteme yüklenen imzanın ayrık-bütünleşik kontrolü yapılması gerekmektedir. İmzalı belge ayrık ise imzalanan içerik istenmelidir. Belge bütünleşik imzalı ise imzalanan içerik istenmemeli fakat kullanıcı içeriği görmek istediğinde bütünleşik imza içerisindeki içerik gösterilmelidir.

9. TÜBİTAK API kullananlar, imza doğrulama kodundaki parametrelere aşağıda belirtilen eklemeleri yapmalıdır:

CAdES	`params.put(EParameters.P_FORCE_STRICT_REFERENCE_USE, true);`

CAdES (Common API) /XAdES	Ortak API için esya-signature-config.xml'de, XadES Native API için xmlsignature-config.xml'de params içerisine aşağıdaki parametre eklenmelidir.
	
	`<!-- loosening below 2 settings will cause warnings instead of validation failure -->`
	`<!-- referenced validation data must be used for cert validation is set true -->`
	`<force-strict-reference-use>true</force-strict-reference-use>`

10. Uygulamada geçersiz imza özet algoritması kontrolü yapılmalıdır. Bu sebeple uygulamanın kabul edilen özet algoritmaları listesi olmalıdır ve bu listede BTK'nın 13 Temmuz 2017 tarihli Resmi Gazete'de yayımlanan "ELEKTRONİK İMZA İLE İLGİLİ SÜREÇLERE VE TEKNİK KRİTERLERE İLİŞKİN TEBLİĞDE DEĞİŞİKLİK YAPILMASINA DAİR TEBLİĞ"nin 1. maddesinde belirtilen özetleme algoritmaları bulunmalıdır. "Uygulama, doğrulama aşamasında imza algoritması içerisinde izin verilen algoritmalar dışında bir algoritma tespit ettiğinde, eğer imza arşivlenmemişse, "Geçersiz Özet Algoritması, İmza En Kısa Sürede Arşivlenmelidir" uyarısı dönmelidir. İmzanın statüsü geçerliyse, özet algoritması uyarısı sebebiyle geçersiz hale getirilmemelidir.

11. İmzacı sertifikası içeriğinde maddi limit bilgisi varsa aşağıdaki yöntemlerden biri izlenerek imza kontrol edilmelidir:
    a. Sertifikada bulunan maddi limit ile belgenin maddi içeriğinin karşılaştırılmadığı durumda, kullanıcıya imzacı sertifikasında maddi limit bilgisi olduğu uyarısı verilmeli ve imzanın doğrulanıp doğrulanmayacağı EBYS uygulaması politikalarına göre belirlenmelidir.
    b. Sertifikada bulunan maddi limit ile belgenin maddi içeriğinin karşılaştırıldığı durumda:
        i. Sertifika maddi limiti belge maddi değeri için yeterli ise imza doğrulanmalıdır.
        ii. Sertifika maddi limiti belge maddi değeri için yeterli değil ise imza doğrulanmamalıdır.

5 Uyum Değerlendirme Test Süreci Hakkında Bilgilendirmeler

Değerlendirmeye alınan kuruluşlar tarafından, yukarıda belirtilen maddelerin uyum değerlendirme çalışması öncesi tamamlanması gerekmektedir. Bu çalışmalar sonrasında uyum değerlendirme süreci başlayacaktır. Bu maddeler uyum değerlendirme sürecinin temel maddeleridir, değerlendirme sürecinde bu maddeler dışında farklı kontroller de yapılacaktır.

Ön Hazırlık çalışması tamamlandıktan sonra kuruma randevu tarihi verilir. Randevu tarihi uyum değerlendirme hizmetinin resmi olarak başladığı tarihtir. Test süreci ön hazırlık tarihinden itibaren en fazla 50 iş günü içerisinde tamamlanmış olmalıdır, aksi takdirde sürecin bedeli kurum tarafından ödenerek yeniden başlatılması gerekmektedir.

EBYS Uyum Değerlendirme testleri dört ana bölümden oluşmaktadır. İlk bölüm olan Format Testi'nde, uygulama kullanılarak oluşturulmuş imzalı dosyaların ilgili standartlarda belirtilen imza formatlarına uygunluğu kontrol edilir. Format Testi'nden sonraki bölüm olan İmza Oluşturma Testi'nde, sertifika ve zaman damgası doğrulama ile ilgili kontroller yapılarak güvenli elektronik imza oluşturma yazılımının uygunluğu test edilir. İmza Doğrulama Testi'nde imza doğrulama testleri yapılarak güvenli elektronik imza doğrulama yazılımının uygunluğu test edilir. Son bölüm olan İmza Arşivleme Testi'nde ise arşivleme uygulamasının değerlendirilmesi yapılmıştır.

Yukarıda ana bölümleri belirtilen Uyum Değerlendirme test maddelerinin uygulamada sağlanmadığı tespit edilirse, kuruma eksikliklerini tamamlaması, uygulamadaki hataları düzeltmesi için zaman tanınır. Kurum hataları düzelttiğini belirttikten sonra tekrar randevu tarihi verilerek test sürecine devam edilir. Test yapılan uygulama, EBYS Uyum Değerlendirme Raporu'nda belirtilen zorunlu olarak sağlanması gereken maddelerin tamamını sağlayana kadar, kuruma hataları iletme, randevu verme ve test sürecine devam etme işlemleri tekrarlanır. EBYS Uyum Değerlendirme Raporu'nda belirtilen tüm zorunlu maddeler sağlandığında, Uyum Değerlendirme testleri tamamlanmış olur.

Uyum Değerlendirme testleri tamamlandıktan sonra uygulamanın arayüz, politika, imza oluşturma ve imza doğrulama kısımlarına ait SHA-256 özet değerlerinin 1 (bir) iş günü içinde, Uyum Değerlendirme Taahhütname Formu'nda belirtilmesi ve bu formun ıslak imzalı ve kaşeli halinin tarafımıza iletilmesi gerekmektedir. Aksi takdirde süreç yenilenecektir. Kurumun Uyum Değerlendirme Taahhütname Formu'nda belirtmiş olduğu özet değerleri, Uyum Değerlendirme Raporu'na yazılır ve raporun onaylanmasının ardından www.kamusm.gov.tr internet sitesinden ilan edilir.

6 Ek-A İmza Arşivleme Rehberi

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 iptal olma ve 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.

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.");
}
```

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.