راهنمای سرویس PKA-DSS

شرکت پندار کوشک ایمن

واحد امنیت اطلاعات و زیرساخت کلید عمومی

نسخه: ۳.۵

تاریخ: تابستان ۱۴۰۴

شناسه: PKI-DSS-API-DG

طبقه‌بندی: عمومی

تاریخچه گزارش

نسخه	تاریخ	تهیه‌کننده/گان	توضیحات
۱.۰	۱۳۹۸/۱۰/۲۳	کارشناس پروژه	نسخه اول جهت انتشار عمومی
۲.۰	۱۳۹۹/۰۴/۲۵	کارشناس پروژه	اضافه شدن سرویس های VAClient, TSAClient ,DSClient
۲.۱	۱۳۹۹/۰۵/۲۵	کارشناس پروژه	اضافه شدن سرویس SignRSA
۲.۲	۱۳۹۹/۰۸/۰۲	کارشناس پروژه	اضافه شدن سرویس‌ها و کنترلرهای Crypto, Login, PKD
۲.۳	۱۴۰۱/۰۴/۲۶	کارشناس پروژه	تغییر نام توابع برخی توابع، حذف توابع اضافی، افزودن توابع جدید
۲.۴	۱۴۰۱/۰۵/۰۲	کارشناس پروژه	افزودن توابع ApiVersion
۲.۵	۱۴۰۱/۰۵/۰۴	کارشناس پروژه	افزودن توابع زیر CmsExtractAttachedMessage, CmsExtractAttachedMessageRaw
۲.۶	۱۴۰۱/۰۵/۱۹	کارشناس پروژه	افزودن توابع زیر: CertificateThumbprint CertificateThumbprintRaw PDFVerifyAndValidateCertificateByKeyUsages PDFVerifyAndValidateCertificateByKeyUsagesRaw ValidateBasicConstraints ValidateBasicConstraintsRaw CmsDigest CmsDigestRaw PutCMSSignature PutCMSSignatureRaw تغییر ترتیب بررسی Ocsp وCrl در تابع ValidateCertificateEntirley
۲.۷	۱۴۰۱/۰۶/۰۵	کارشناس پروژه	افزودن توابع زیر: ValidateCertificateByCRLFull, ValidateCertificateByCRLFullRaw, ValidateCertificateByOCSPFull, ValidateCertificateByOCSPFullRaw حذف توابع زیر:

			PDFVerifyAndValidateCertificateByKeyUsages PDFVerifyAndValidateCertificateByKeyUsagesRaw
۲.۸	۱۴۰۱/۰۶/۱۳	کارشناس پروژه	تغییر نام توابع ذیل: کلمه Full به Ex تغییر داده شد. ValidateCertificateByCRLEx, ValidateCertificateByCRLExRaw, ValidateCertificateByOCSPEx, ValidateCertificateByOCSPExRaw افزودن تابع ValidateCertificateEntirelyEx ValidateCertificateEntirelyExRaw
۲.۹	۱۴۰۱/۰۶/۱۵	کارشناس پروژه	افزودن توابع Sign , SignRaw
۲.۱۰	۱۴۰۱/۰۹/۱۹	کارشناس پروژه	افزودن مقادیر HashAlgorithm
۲.۱۱	۱۴۰۱/۰۹/۲۰	کارشناس پروژه	افزودن پارامتر AuthRequest به توابع کلاس TsaService , DsService برای اتصال به Sign سرورها
۲.۱۲	۱۴۰۲/۰۵/۱۵	کارشناس پروژه	افزودن ۴ api جدید ذیل به سرویس Crypto: - SignByP12Raw - SignByP12 - PdfMultiSignByP12Raw - PdfMultiSignByP12
۲.۱۳	۱۴۰۲/۰۷/۱۶	کارشناس پروژه	افزودن ۲ api جدید ذیل به سرویس Crypto: - MergeCmsAttachSign - MergeCmsAttachSignRaw
۲.۱۴	۱۴۰۳/۰۴/۰۲	کارشناس پروژه	- بروزرسانی توابع پرکاربرد - بهبود ورودی و خروجی‌های توابع پرکاربرد
۲.۱۵	۱۴۰۳/۰۴/۱۳	کارشناس پروژه	- بروزرسانی توضیحات امضای سند PDF
۲.۱۶	۱۴۰۳/۰۹/۲۱	کارشناس	- بروزرسانی توابع - بهبود ورودی و خروجی‌های توابع
۲.۱۷	۱۴۰۳/۱۱/۲۱	کارشناس	- بهبود ورودی توابع Sign , SignRaw

۲.۱۸	۱۴۰۳/۱۱/۲۱	کارشناس	- افزودن توابع CMSSign , CMSSignRaw
۲.۱۹	۱۴۰۳/۱۲/۰۱	کارشناس	- افزودن توابع MergeCmsSign, MergeCmsSignRaw
۳.۰	۱۴۰۴/۰۳/۱۸	کارشناس	- افزدون Version number به ورودی توابع - مشخص نمودن توابع Deprecate شده
۳.۲	۱۴۰۴/۰۴/۰۱	کارشناس	- افزودن نمودار توالی فرآیندها
۳.۳	۱۴۰۴/۰۴/۰۱	کارشناس	- بازنگری و ویرایش
۳.۴	۱۴۰۴/۰۵/۰۴	کارشناس	- بازنگری و ویرایش نمودار توالی فرآیندها
۳.۵	۱۴۰۴/۰۶/۰۹	کارشناس	- بازنگری توضیحات متد PdfMultiSignByP12

فهرست مطالب

۱ مقدمه

در این مستند راهنمایی‌های لازم جهت استفاده از توابع سرویس PKA به منظور انجام عملیات‌های اعتبارسنجی گواهی‌نامه‌ها و امضای اسناد و مهر زمانی و همچنین عملیات‌های صدور گواهینامه و همچنین ابطال گواهینامه و نیز تعلیق و فعال‌سازی گواهینامه ارائه شده است. خدماتی که این دسته از توابع ارائه می‌دهند، در ادامه مورد بررسی قرار خواهد گرفت. .

همچنین این راهنما به صورت اختصاصی برای برنامه‌نویسان و توسعه‌دهندگان نرم‌افزار تهیه شده است و قدر مسلم نیاز به دانش‌های اولیه برنامه‌نویسی دارد.

در نسخه شماره ۳.۰ از سامانه DSS قابلیت نسخه گذاری به سرویس ها افزوده شده است که در ادامه به شرح آن می پردازیم. در نسخه پیش رو از سامانه ۲ نسخه متفاوت از متدها در دسترس است که عبارتند از نسخه شماره ۲.۰ و نسخه شماره ۳.۰، در نسخه شماره ۲.۰ متدها و سرویس ها هیچ تغییر محسوسی برای کاربر ندارند ولی در نسخه شماره ۳.۰ متد ها و سرویس هایی که آدرس آنها به کلمه Raw ختم می شدند حذف گردیده است. برای استفاده از قابلیت نسخه بندی کاربر باید مقدار شماره نسخه مورد نظر خود را وارد کند در غیر این صورت نسخه پیش فرض که در اینجا نسخه ۲.۰ می باشد به کار گرفته خواهد شد. برای استفاده از این قابلیت کاربر مقدار متقارن با نسخه مورد خود را در header درخواست ارسال کند و در هدر پاسخ به درخواست نیز می تواند مشاهده کند که سرویس را از چه نسخه ای گرفته است، کدام نسخه منسوخ شده و کدام نسخه پشتیبانی می شود. به این منظور کاربر باید در تمام درخواست ها مقدار شماره نسخه مورد نظر خود را در پارامتر api-version در header درخواست مشخص و ارسال کند. همانگونه که توضیح داده شد در صورت عدم مشخص کردن شماره نسخه مورد نظر، API های نسخه ۲ که در حال حاضر نسخه پیش فرض هستند فراخوانی می شود.

۲ سرویس اعتبارسنجی(VAService)

سرویس VAService به منظور دسته‌بندی توابع مربوط به بررسی صحت اعتبار گواهینامه در نظر گرفته شده است. توابع موجود در این سرویس، استانداردهای مختلف دریافت وضعیت یک گواهینامه را پیاده‌سازی می‌نماید. خدماتی که این دسته از توابع ارائه می‌دهد، در ادامه مورد بررسی قرار خواهد گرفت.

۲/۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

<Http protocol>://DSSUrl/API/VAService

۲/۲ بررسی اعتبار امضای دیجیتال و تاریخ اعتبار گواهینامه

تابعی که این خدمت را ارائه می‌کند، گواهی ارائه‌شده را از نظر اعتبار آن در زمان کنونی و همچنین امضای دیجیتال صادرشده بر روی آن از طرف CA صادرکننده‌ی آن، مورد ارزیابی قرار می‌دهد.

جزئیات متد ارائه‌دهنده‌ی این خدمت در سرویس VAService به صورت زیر است:

api/VAService/ValidateCertificate	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	base64: string گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه اعتبار سنجی گواهی

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

۲/۳ بررسی اعتبار گواهینامه با سرویس استعلام آنلاین وضعیت گواهینامه (OCSP)

api/VAService/ValidateCertificateByOCSPEx	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateRequest<string>
	{
	"certificate": "string",
	گواهی موردنظر جهت اعتبارسنجی در قالب Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به آدرس OCSP
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	پاسخ سرویس OCSP
	{ "error": "string", "result": OCSPResponseStatus<string> { "certificateStatus": certificateStatus, جدول شماره ۱

	نتیجه اعتبارسنجی آنلاین گواهی "revocationTime": dateTime, زمان ابطال گواهی[درصورت باطل بودن گواهی] "revocationReason": RevocationReason, جدول شماره ۲ علت ابطال گواهی [درصورت باطل بودن گواهی] "certificate": "string" گواهی اعتبارسنجی شده در قالبBase64 }, "statusCode": number => 200 Success / 500 Error }

api/VAService/ValidateCertificateByOCSP	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateRequest<string>
	{
	"certificate": "string",
	گواهی موردنظر جهت اعتبارسنجی در قالب Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به آدرس OCSP
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه اعتبارسنجی آنلاین گواهی

	{
	"error": string,
	"result": CertificateStatus, جدول شماره ۱
	"statusCode": number => 200 Success / 500 Error
	}

api/VAService/ValidateCertificateListByOCSP	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0
Schema	ChainValidationRequest<string>

	{
	"certificate": [
	"string"
	],
	فهرستی از گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به آدرس OCSP
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از نتایج استعلام آنلاین گواهی

	{
	"error": "string",
	"result": IEnumerable<OCSPResponseStatus<string>
	[
	{
	"certificateStatus": CertificateStatus, جدول شماره ۱
	نتیجه اعتبارسنجی آنلاین گواهی
	"revocationTime": dateTime,
	زمان ابطال گواهی [درصورت باطل بودن گواهی]
	"revocationReason": RevocationReason, جدول شماره ۲
	علت ابطال گواهی[درصورت باطل بودن گواهی]
	"certificate": string
	گواهی اعتبارسنجی شده با فرمت Base64
	}
	],
	"statusCode": number => 200 Success / 500 Error
	}

باتوجه به اختیاری بودن پارامتر vaProfile، ارسال آن توصیه نمی‌گردد. درصورت لزوم، جهت اطلاع از نحوه مقداردهی به پارامتر vaProfile با تیم استقرار سرویس همانگ شوید.

۲/۴ بررسی اعتبار گواهینامه(ها) با لیست گواهی باطله(CRL)

api/VAService/ValidateCertificateByCRLEx	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain

	api-version : 2.0 or 3.0

Schema	ValidateRequest<string>
	{
	"certificate": "string",
	گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به مسیر فایل CRL
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه بازگشتی از بررسی CRL

	{
	"error": "string",
	"result": CRLRevocationResult<string>
	{
	"isValid": bool,
	نتیجه اعتبارسنجی آفلاین گواهی
	"revocationTime": "dateTime",
	زمان ابطال گواهی [درصورت باطل بودن گواهی]
	"certificate": "string"
	گواهی اعتبارسنجی شده با فرمت Base64
	},
	"statusCode": number => 200 Success / 500 Error
	}

api/VAService/ValidateCertificateByCRL	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateRequest<string>
	{
	"certificate": "string",
	گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به مسیر فایل CRL
	}

Response	

Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه اعتبارسنجی آفلاین گواهی

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

api/VAService/ValidateCertificateListByCRL	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateRequest<string>
	{
	"certificate": [
	"string"
	],
	فهرستی از گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به مسیر فایل CRL
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از نتایج بازگشتی از بررسی CRL

	{
	"error": "string",
	"result": CRLRevocationResult<string>
	[
	{
	"isValid": bool,
	نتیجه اعتبارسنجی آفلاین گواهی
	"revocationTime": "dateTime",
	زمان ابطال گواهی [درصورت باطل بودن گواهی]
	"certificate": "string"
	گواهی اعتبارسنجی شده با فرمت Base64
	}
	],

	"statusCode": number => 200 Success / 500 Error }

باتوجه به اختیاری بودن پارامتر vaProfile، ارسال آن توصیه نمی‌گردد. درصورت لزوم، جهت اطلاع از نحوه مقداردهی به پارامتر vaProfile با تیم استقرار سرویس همانگ شوید.

۲٫۵ بررسی اعتبار گواهینامه با کاربردهای گواهی و کاربردهای توسعه یافته آن

api/VAService/ValidateCertificateByKeyUsage	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateRequest<string>
	{
	"certificate": "string",
	گواهی موردنظر جهت بررسی کاربردهای آن با فرمت Base64
	"vaProfile": "string" توجه
	نام پروفایل مورد استفاده جهت دسترسی به کاربردهای موردنیاز
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه ارزیابی گواهی

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

در این متد با استفاده از فهرست KeyUsages و ExtendedKeyUsage هایی که در تگ مربوطه در vaProfile مشخص شده است، کاربردهای گواهی و کاربردهای توسعه‌یافته گواهی بررسی می‌گردد. لازم به توضیح است که :

۱. در تگ KeyUsages، با توجه به نیاز، می‌توان یک یا چندین مورد از اقلام موجود در جدول شماره ۷ را قرار داد.
کاراکتر جداکننده در این لیست علامت | می‌باشد. به‌طور مثال:

KeyUsages="DIGITALSIGNATURE|NONREPUDIATION" یا **KeyUsages**="6|7"

۲. در تگ ExtendedKeyUsage، باید OID هر یک از کاربردها توسعه‌یافته برحسب نیاز قرارداده شود.
کاراکتر جداکننده در این لیست علامت | می‌باشد. به‌طور مثال:

ExtendedKeyUsage="1.2.840.113583.1.1.5|1.3.6.1.5.5.7.3.1"

٣. جهت اطلاع از نحوه مقداردهی به پارامتر vaProfile با تیم استقرار سرویس همانگ شوید.

api/VAService/ValidateCertificateByKeyUsageEx	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidationByKeyUsageRequest<string>
	{
	"certificate": "string",
	گواهی موردنظر جهت بررسی کاربردهای آن با فرمت Base64
	"keyUsages": [
	IEnumerable<KeyUsage> توجه ١
	],
	فهرستی از کاربردهای مورد نیاز برای بررسی
	"extendedKeyUsages": [
	IEnumerable<string> توجه ٢
	]
	فهرستی از old های مورد نیاز برای بررسی
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه ارزیابی گواهی

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

١. کاربردهای گواهی با استفاده از فهرستی که در فیلد KeyUsages مشخص شده است، بررسی می‌گردد. لازم به توضیح است که در فیلد KeyUsages، با توجه به نیاز، می‌توان یک یا چندین مورد از اقلام موجود در جدول شماره ٧ را قرار داد. کاراکتر جداکننده در این فهرست علامت , می‌باشد. به‌طور مثال:

```
"keyUsages": [
  6,7
 ]
```

۲. کاربردهای توسعه‌یافته گواهی با استفاده از فهرستی که در فیلد ExtendedKeyUsage، مشخص شده است بررسی می‌شوند. توجه شود که در فیلد ExtendedKeyUsage، باید OID هر یک از کاربردها توسعه‌یافته برحسب نیاز قرارداده شود.

کاراکتر جداکننده در این لیست علامت , می‌باشد. به‌طور مثال:

```
"extendedKeyUsages": [
  "1.3.6.1.5.5.7.3.2","1.3.6.1.5.5.7.3.1"
 ]
```

درصورت عدم نیاز به بررسی KeyUsage و یا ExtendedKeyUsage، می بایست پارامتر مربوطه بدون مقدار ارسال شود. به طور مثال:

```
"extendedKeyUsages": [
 ]
```

۲/۶ بررسی کلی اعتبار گواهینامه

api/VAService/ValidateCertificateEntirelyEx	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateEntirelyRequest <string>
	{
	"certificate": "string",
	گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
	"vaProfile": "string" اختیاری
	نام پروفایل مورد استفاده جهت دسترسی به آدرس های OCSP و CRL و کاربردهای گواهی
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتایج بازگشتی از بررسی کلی اعتبار گواهی
	{ "error": "string", "result": CertificateValidationResult<string> { "certificateValidationStatus": CertificateValidationStatus, جدول شماره ۳ وضعیت گواهی "revocationTime": "dateTime"

	زمان ابطال گواهی [درصورت باطل بودن گواهی] }, "statusCode": number => 200 Success / 500 Error }

api/VAService/ValidateCertificateEntirely	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	ValidateEntirelyRequest <string>
	{
	"certificate": "string",
	گواهی موردنظر جهت اعتبارسنجی با فرمت Base64
	"vaProfile": "string" توجه
	نام پروفایل مورد استفاده جهت دسترسی به آدرس های OCSP و CRL و کاربردهای گواهی
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه بازگشتی از بررسی کلی اعتبار گواهی

	{
	"error": "string",
	"result": CertificateValidationStatus, جدول شماره ۳
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

- جهت اطلاع از نحوه مقداردهی به پارامتر vaProfile با تیم استقرار سرویس هماهنگ شوید.

۲٫۷ دانلود فایل CRL

استاندارد CRL یا Certificate Revocation List که در زمینه‌ی بررسی اعتبار گواهینامه موجود است، نیازمند بررسی سریال گواهینامه با فایل تولید شده توسط CA می‌باشد. متد DownloadCRL جهت دانلود فایل CRL پیاده‌سازی شده است .

جزئیات متد ارائه‌دهنده‌ی این خدمت در سرویس VAService به صورت‌های زیر است:

api/VAService/DownloadCRLByCertificate	
Request	
Method	Post
Header	Content-Type: multipart/form-data

	Accept: text/plain api-version : 2.0 or 3.0
Body	base64: string گواهی مورد نظر جهت دسترسی به آدرس CRL موجود در آن با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فایل CRL دانلود شده با فرمت byte[]

	{
	"error": "string",
	"result": byte[],
	"statusCode": number => 200 Success / 500 Error
	}

۲٫۸ بررسی اقلام استاندارد گواهی‌های میانی و ریشه

api/VAService/ValidateBasicConstraints	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	base64Cert: string گواهی مورد نظر جهت اعتبارسنجی با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه اعتبارسنجی گواهی

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

۳ سرویس امضای اسناد الکترونیک(DSService)

سرویس DSService توابع مورد نیاز جهت امضای اسناد توسط DocumentSigner را ارائه می‌دهند.

در صورتی که گواهی سازمانی در DocumentSigner تنظیم شده باشد، به این عملیات مهرسازمانی گفته میشود

خدماتی که این دسته از توابع ارائه می‌دهند در ادامه مورد بررسی قرار خواهد گرفت.

۳٫۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

<Http protocol>://DSSUrl/API/DSService

۳٫۲ درخواست امضای سند PDF

الزامیست نسخه PDF ۱.۶ به بالاتر باشد.

api/DSService/PDFSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	SignRequest<string>
	{
	"data": "string",
	سند PDF با فرمت Base64
	"dsProfile": "string", توجه ۱
	نام پروفایل جهت ارتباط با DocumentSigner
	"authRequest": "string" توجه ۲
	نام کاربری و رمز عبور جهت ارتباط با DocumentSigner
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	سند PDF امضاشده توسط DocumentSigner با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. جهت اطلاع از نحوه مقداردهی به پارامتر dsProfile با تیم استقرار سرویس هماهنگ شوید.

۲. درصورتی که برقراری ارتباط با DocumentSigner نیازمند احراز هویت می‌باشد، پارمتر authRequest باید مطابق با ساختار زیر مقدارهی شود درغیر این صورت نیاز به مقداردهی این پارامتر نمی‌باشد.

ConvertToBase64(User:Password)

لازم به ذکر است نام کاربری و رمز عبور باید از تیم استقرار دریافت گردد.

۳/۳ درخواست امضای سند CMS

api/DSService/CMSSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CMSSignRequest<string>
	{
	"data":"string",
	محتوای درخواستی جهت امضا با فرمت Base64
	"dsProfile": "string", توجه ۱
	نام پروفایل جهت ارتباط با DocumentSigner
	"authRequest": "string", توجه ۲
	نام کاربری و رمز عبور جهت ارتباط با DocumentSigner
	"attachData": bool
	مقدار boolean جهت انتخاب قراردادن محتوای درخواستی در قالب CMS
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	محتوای امضاشده توسط DocumentSigner با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. جهت اطلاع از نحوه مقداردهی به پارامتر dsProfile با تیم استقرار سرویس هماهنگ شوید.

۲. درصورتی که برقراری ارتباط با DocumentSigner نیازمند احراز هویت می‌باشد، پارمتر authRequest باید مطابق با ساختار زیر مقدارهی شود درغیر این صورت نیاز به مقداردهی این پارامتر نمی باشد.

ConvertToBase64(User:Password)

لازم به ذکر است نام کاربری و رمز عبور باید از تیم استقرار دریافت گردد.

۳/۴ درخواست امضای RSA

api/DSService/RSASign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	SignRequest<string>
	{
	"data": "string",
	محتوای درخواستی جهت امضا با فرمت Base64
	"dsProfile": "string", توجه ۱
	نام پروفایل جهت ارتباط با DocumentSigner
	"authRequest": "string" توجه ۲
	نام کاربری و رمز عبور جهت ارتباط با DocumentSigner
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	محتوای امضاشده توسط DocumentSigner با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. جهت اطلاع از نحوه مقداردهی به پارامتر dsProfile با تیم استقرار سرویس هماهنگ شوید.

۲. درصورتی که برقراری ارتباط با DocumentSigner نیازمند احراز هویت می‌باشد، پارمتر authRequest باید مطابق با ساختار زیر مقدارهی شود درغیر این صورت نیاز به مقداردهی این پارامتر نمی باشد.

ConvertToBase64(User:Password)

لازم به ذکر است نام کاربری و رمز عبور باید از تیم استقرار دریافت گردد.

۳/۵ درخواست امضای سند XML

api/DSService/XMLSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain

	api-version : 2.0 or 3.0

Schema	SignRequest<string>
	{
	"data":"string",
	سند XML درخواستی جهت امضا با فرمت Base64
	"dsProfile": "string", توجه ۱
	نام پروفایل جهت ارتباط با DocumentSigner
	"authRequest": "string" توجه ۲
	نام کاربری و رمز عبور جهت ارتباط با DocumentSigner
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	سند XML امضاشده توسط DocumentSigner با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. جهت اطلاع از نحوه مقداردهی به پارامتر dsProfile با تیم استقرار سرویس هماهنگ شوید.

۲. درصورتی که برقراری ارتباط با DocumentSigner نیازمند احراز هویت می‌باشد، پارمتر authRequest باید مطابق با ساختار زیر مقدارهی شود درغیر این صورت نیاز به مقداردهی این پارامتر نمی باشد.

ConvertToBase64(User:Password)

لازم به ذکر است نام کاربری و رمز عبور باید از تیم استقرار دریافت گردد.

۴ سرویس مهر زمانی مطمئن (TSAService)

سرویس TSAService توابع مورد نیاز جهت ارسال اطلاعات مورد نیاز جهت صدور مهر زمانی مطمئن بر روی آن به سرور TSA و همچنین دریافت پاسخ دریافتی از سرور را ارائه می‌دهند. خدماتی که این دسته از توابع ارائه می‌دهند در ادامه مورد بررسی قرار خواهد گرفت.

۴/۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

<Http protocol>://DSSUrl/API/ TSAService

۴/۲ درخواست مهر زمانی مطمئن

api/TSAService/TSTSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	TSTSignRequest<string>
	{
	"message": "string",
	محتوای درخواستی جهت صدور مهر زمانی مطمئن بر روی آن با فرمت Base64
	"requestCertificate": bool,
	مقدار boolean جهت انتخاب قرارگیری گواهی سرور TSA در خروجی
	"tsaProfile": "string", توجه ۱
	نام پروفایل جهت ارتباط با سرور TSA
	"authRequest": "string" توجه ۲
	نام کاربری و رمز عبور جهت ارتباط با سرور TSA
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	مهر زمانی صادرشده برروی اطلاعات درخواستی با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. جهت اطلاع از نحوه مقداردهی به پارامتر tsaProfile با تیم استقرار سرویس هماهنگ شوید.

۲. درصورتی که برقراری ارتباط با سرور TSA نیازمند احراز هویت می‌باشد، پارمتر authRequest باید مطابق با ساختار زیر مقدارهی شود درغیر این صورت نیاز به مقداردهی این پارامتر نمی باشد.

ConvertToBase64(User:Password)

لازم به ذکر است نام کاربری و رمز عبور باید از تیم استقرار دریافت گردد.

۵ سرویس رمزنگاری و رمزگشایی(CryptoService)

CryptoService توابع لازم برای رمزنگاری و رمزگشایی را در قالب یک وب سرویس ارائه می‌دهد. خدماتی که این دسته از توابع ارائه می‌دهند در ادامه مورد بررسی قرار خواهد گرفت.

۵/۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

<Http Protocol>://DSSUrl/API/ CryptoService

۵/۲ مبدل رشته Base۶۴ به Unicode

api/CryptoService/Base64ToUnicode	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	base64String: string رشته موردنظر با فرمت Base64 برای تبدیل به فرمت Unicode
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	رشته ورودی به فرمت Unicode

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۵/۳ مبدل رشته Unicode به Base۶۴

api/CryptoService/UnicodeToBase64	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	unicode: string رشته موردنظر با فرمت Unicode برای تبدیل به فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	رشته ورودی به فرمت Base64
	{ "error": "string", "result": "string",

	"statusCode": number => 200 Success / 500 Error }

۵٫۴ استخراج گواهی از قالب CMS

api/CryptoService/CMSExtractCertificates	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	envloped : string محتوای CMS با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از گواهی های استخراج شده با فرمت Base64

	{
	"error": "string",
	"result": IEnumerable<string>,
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۵ تصدیق امضای دیجیتال در قالب CMS

api/CryptoService/CMSVerifyAttach	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	envloped : string محتوای امضاشده (CMS) با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی امضا

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

api/CryptoService/CMSVerify	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CmsVerifyRequest <string>
	{
	"signature": "string",
	محتوای امضاشده (CMS) با فرمت Base64
	"message": "string"
	محتوای اولیه (قبل از امضا) با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی امضا

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۶ تصدیق امضای دیجیتال در قالبCMS و اعتبار سنجی گواهی

api/CryptoService/CmsVerifyAndValidateCertificateAttach	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Body	PdfVerifyAndValidateRequest<byte[]>
	{
	"signedData": "string",
	محتوای امضاشده (CMS) با فرمت byte[]
	"vaProfile": "string" توجه
	نام پروفایل موردنظر جهت استفاده در عملیات اعتبارسنجی گواهی با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی امضا

	{
	"error": "string",
	"result": IEnumerable<VerificationResult>
	[
	{
	"certificate": "string",
	گواهی امضا با فرمت Base64
	"status": CMSVerifyValidationStatus, جدول شماره ۵
	وضعیت امضا
	"revision": number,
	شمارنده ی امضا
	"result": bool
	نتیجه ارزیابی امضا
	}
	],
	"statusCode": number => 200 Success / 500 Error
	}

api/CryptoService/CmsVerifyAndValidateCertificate	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CmsVerifyAndValidateRequest<string>
	{
	"signature": "string",
	محتوای امضاشده (CMS) با فرمت Base64
	"message": "string",
	محتوای اولیه (قبل از امضا) با فرمت Base64
	"vaProfile": "string" توجه
	نام پروفایل موردنظر جهت استفاده در عملیات اعتبارسنجی گواهی با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از نتایج بازگشتی از بررسی امضا
	{ "error": "string", "result": IEnumerable<VerificationResult>

	[
	{
	"certificate": "string",
	گواهی امضا با فرمت Base64
	"status": CMSVerifyValidationStatus, جدول شماره ۵
	وضعیت امضا
	"revision": number,
	شمارنده ی امضا
	"result": bool
	نتیجه ارزیابی امضا
	}
	],
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

- جهت اطلاع از نحوه مقداردهی به پارامتر vaProfile با تیم استقرار سرویس هماهنگ شوید.

۵/۷ دریافت Policyهای گواهینامه

api/CryptoService/CertificatePolicies	
Request	
Method	Post
Header	Content-Type: application/octet-stream Accept: text/plain api-version : 2.0 or 3.0
Body	base64Data: string گواهینامه موردنظر با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	لیست Policy های گواهینامه

	{
	"error": "string",
	"result": IEnumerable<string>,
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۸ ایجاد Digest برای امضای یک پیام

api/CryptoService/Digest	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Body	DigestRequest<string>
	{
	"message": "string",
	متن پیام با فرمت Base64
	"hashAlgorithm": HashAlgorithm جدول شماره ۴
	الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضای پیام
	}

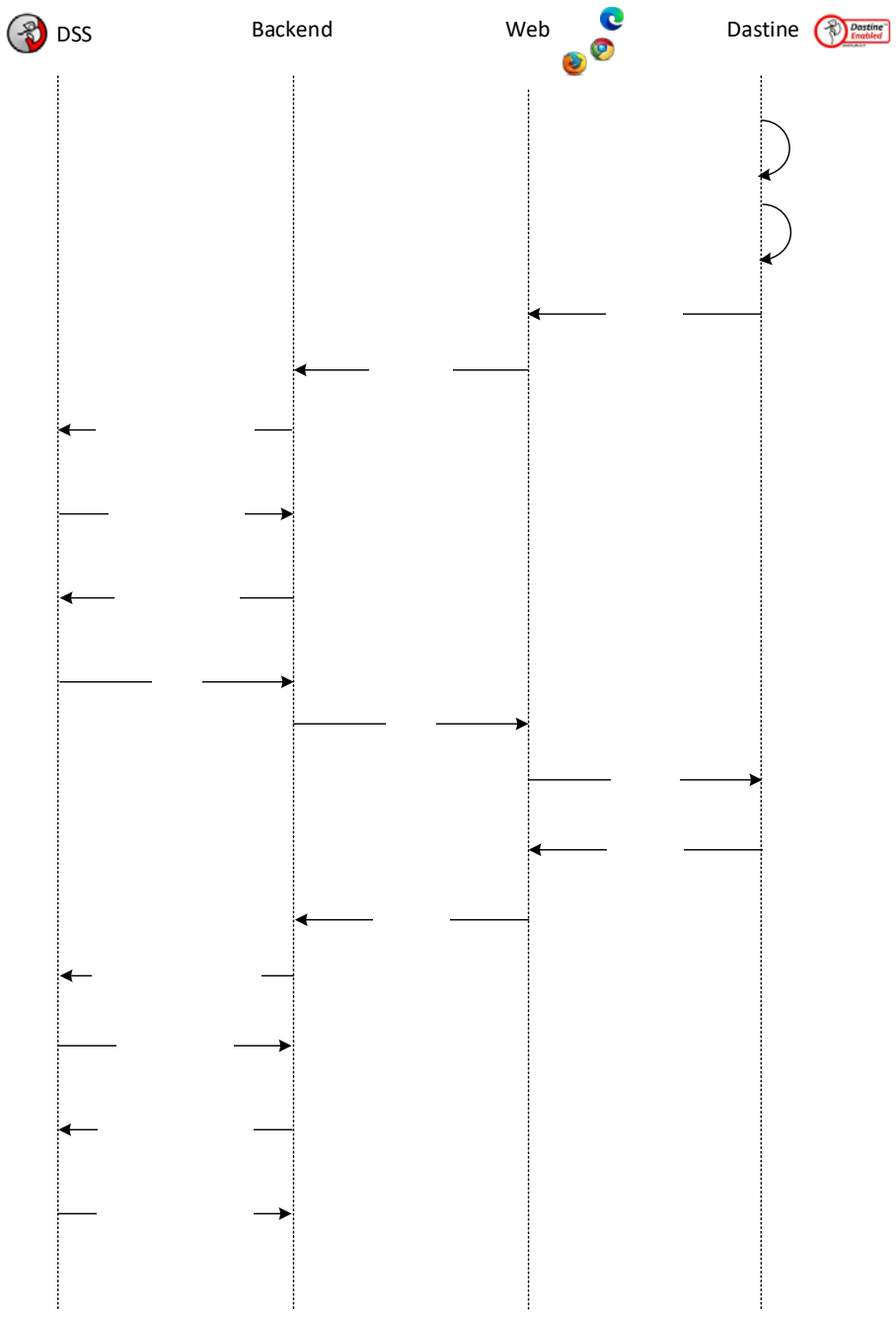
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	Digest پیام با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

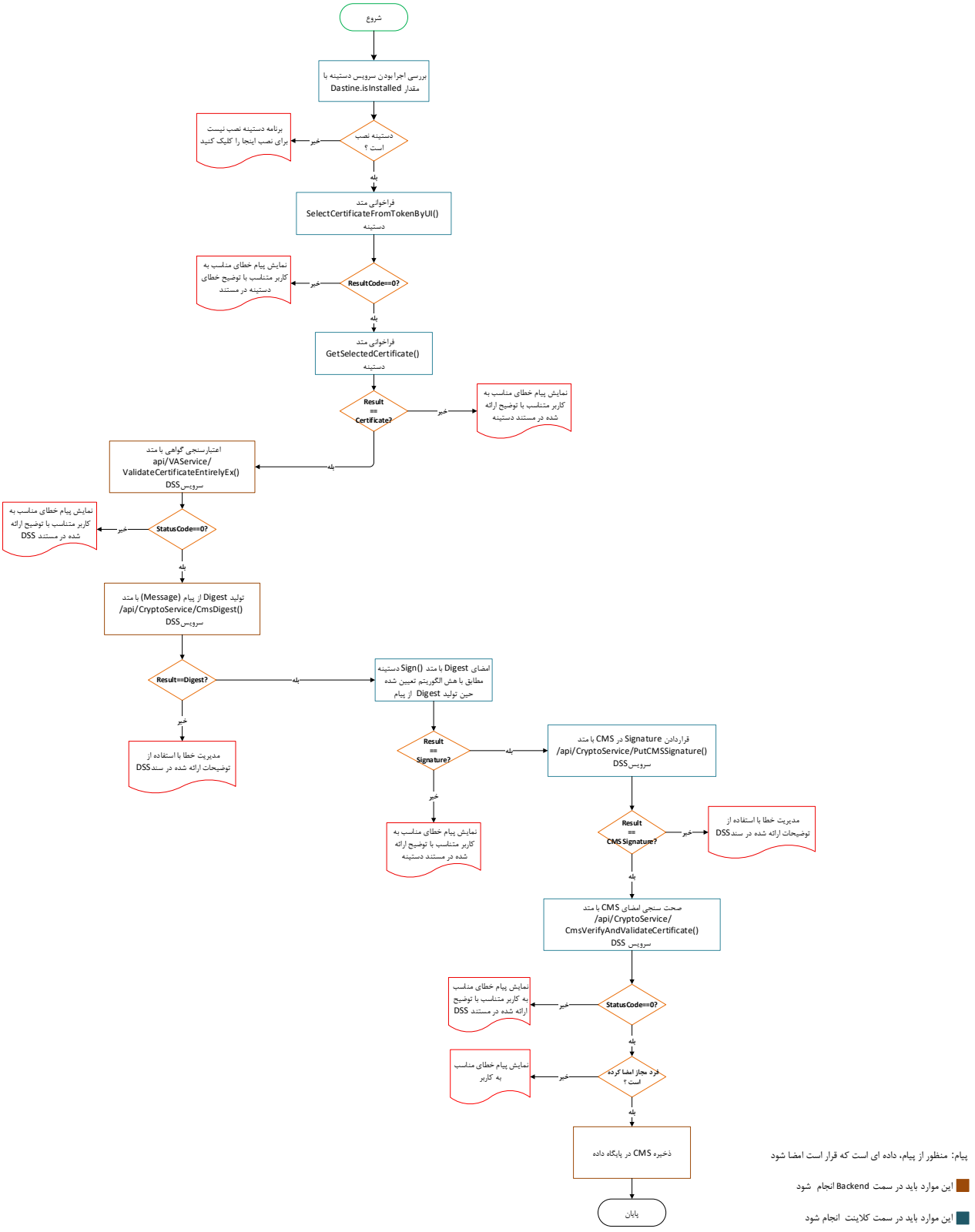
۵/۹ امضای پیام در قالب CMS

۵/۹/۱ امضای **CMS** توسط دستینه

نمودار توالی امضای CMS توسط دستینه



فلوچارت توالی امضای CMS توسط دستینه

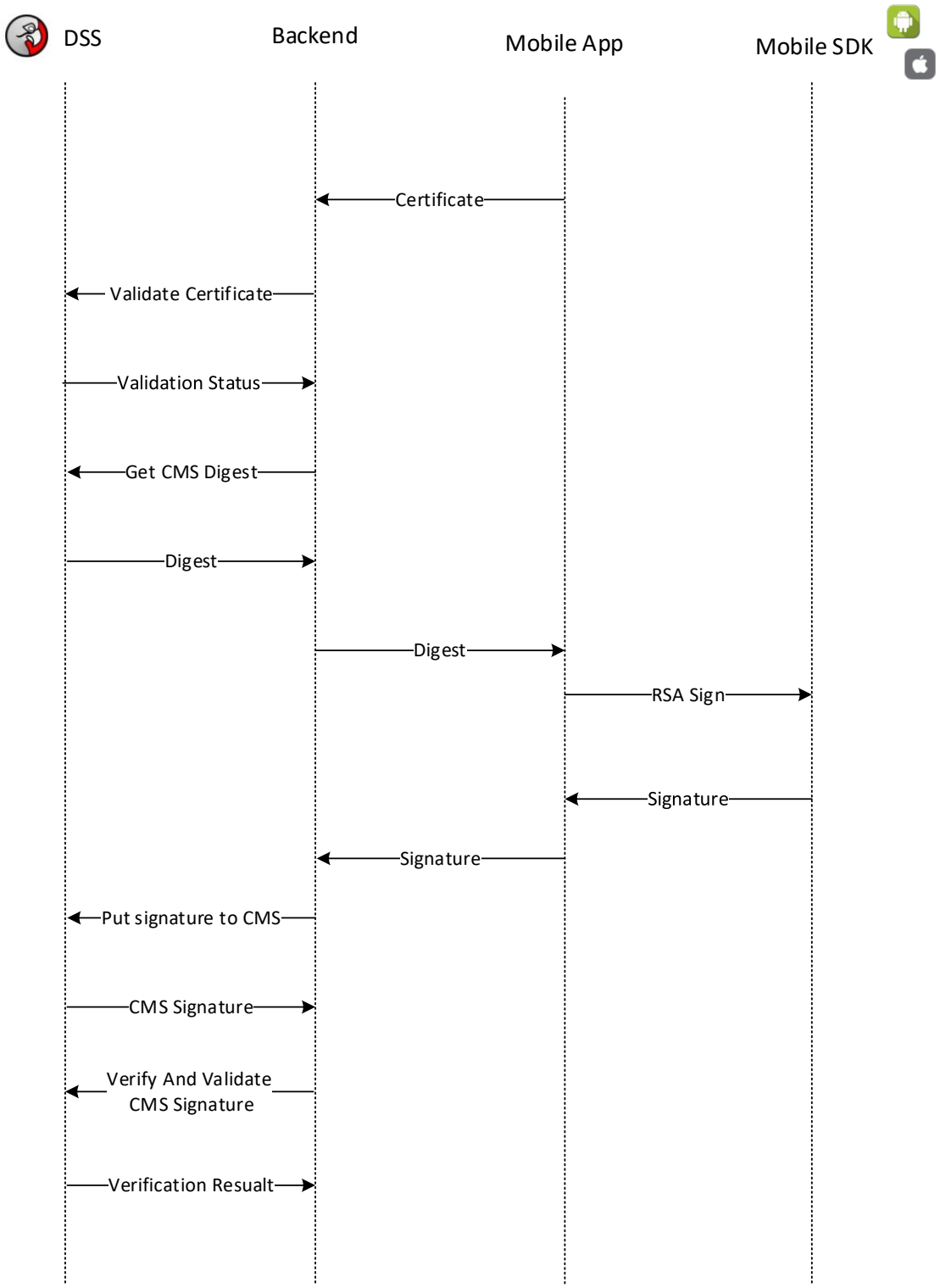


فهرست توابع موردنیاز جهت امضای CMS با دستینه:

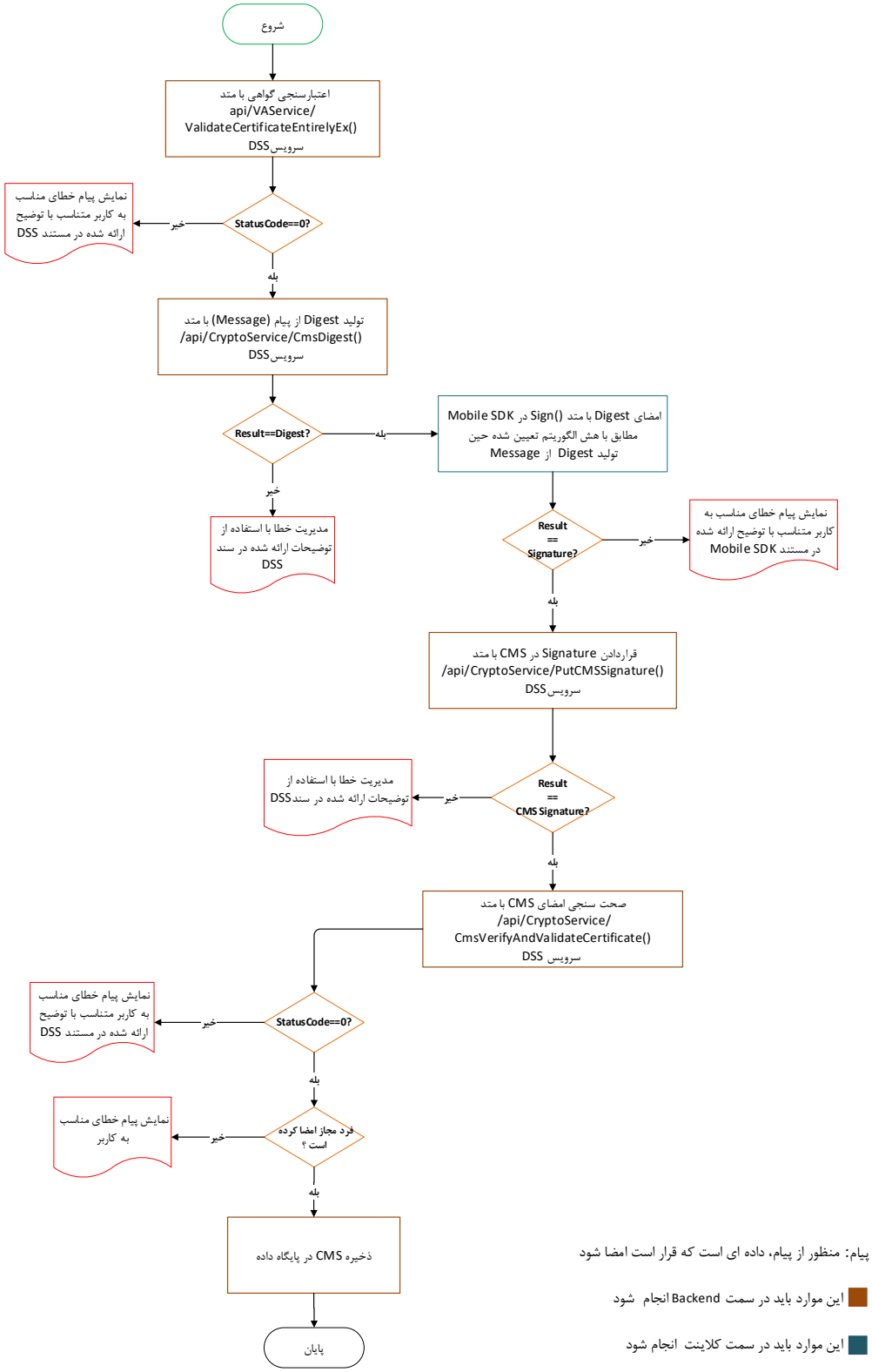
۱. انتخاب گواهی موردنظر جهت امضا با فراخوانی متد SelectCertificateFromTokenByUI() در دستینه
۲. دریافت گواهی انتخاب شده با فرمت Base64 توسط متد GetSelectedCertificate() در دستینه
۳. اعتبارسنجی گواهی از طریق متد /api/VAService/ValidateCertificateEntirelyEx در سرویس DSS
۴. تولید Digest از پیام با استفاده از متد /api/CryptoService/CmsDigest در سرویس DSS
۵. استفاده از متد Sign() دستینه با پارامترهای ورودی زیر جهت امضای Digest تولید شده
   - string data : Didgest تولید شده با فرمت Base64
   - String hashAlg : الگوریتم هش استفاده شده حین تولید Digest از پیام
۶. قراردادن امضا در CMS با فراخوانی متد /api/CryptoService/PutCMSSignature در سرویس DSS
۷. صحت سنجی امضای CMS با فراخوانی یکی از متدهای زیر در سرویس DSS
   - درصورتی که امضا از نوع CMSAttach میباشد از متد
     /api/CryptoService/CmsVerifyAndValidateCertificateAttach در سرویس DSS استفاده نمایید
   - درصورتی که امضا از نوع CMSDetach میباشد از متد
     /api/CryptoService/CmsVerifyAndValidateCertificate در سرویس DSS استفاده نمایید

۵/۹/۲ امضای **CMS** توسط **SDK** موبایل

نمودار توالی امضای CMS توسط SDK موبایل



فلوچارت توالی امضای CMS توسط SDK موبایل



فهرست توابع موردنیاز جهت امضای CMS توسط SDK موبایل:

۱. اعتبارسنجی گواهی امضا از طریق متد /api/VAService/ValidateCertificateEntirelyEx در سرویس DSS

۲. تولید Digest از پیام با استفاده از متد /api/CryptoService/CmsDigest در سرویس DSS

۳. استفاده از متد ()Sign در Mobile SDK با پارامترهای ورودی زیر جهت امضای Digest تولید شده :

- string message : Didgest تولید شده با فرمت Base64
- string privateKeyName : نام کلید خصوصی (مطابق با توضیحات ارائه شده در سند SDK موبایل مقداردهی شود)
- string hashAlg : الگوریتم هش استفاده شده حین تولید Digest از پیام
- string pin : پین کلید خصوصی (مطابق با توضیحات ارائه شده در سند SDK موبایل مقداردهی شود)

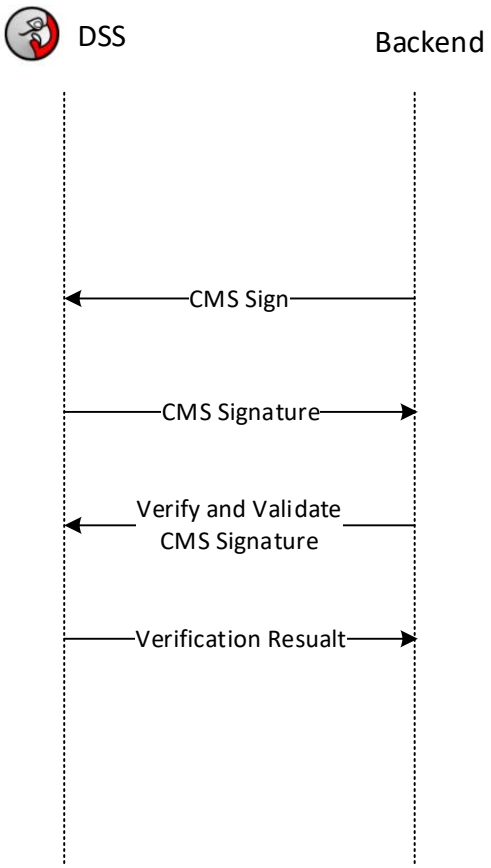
٤. قراردادن امضا در CMS با فراخوانی متد /api/CryptoService/PutCMSSignature در سرویس DSS

٥. صحت سنجی امضای CMS با فراخوانی یکی از متدهای زیر در سرویس DSS

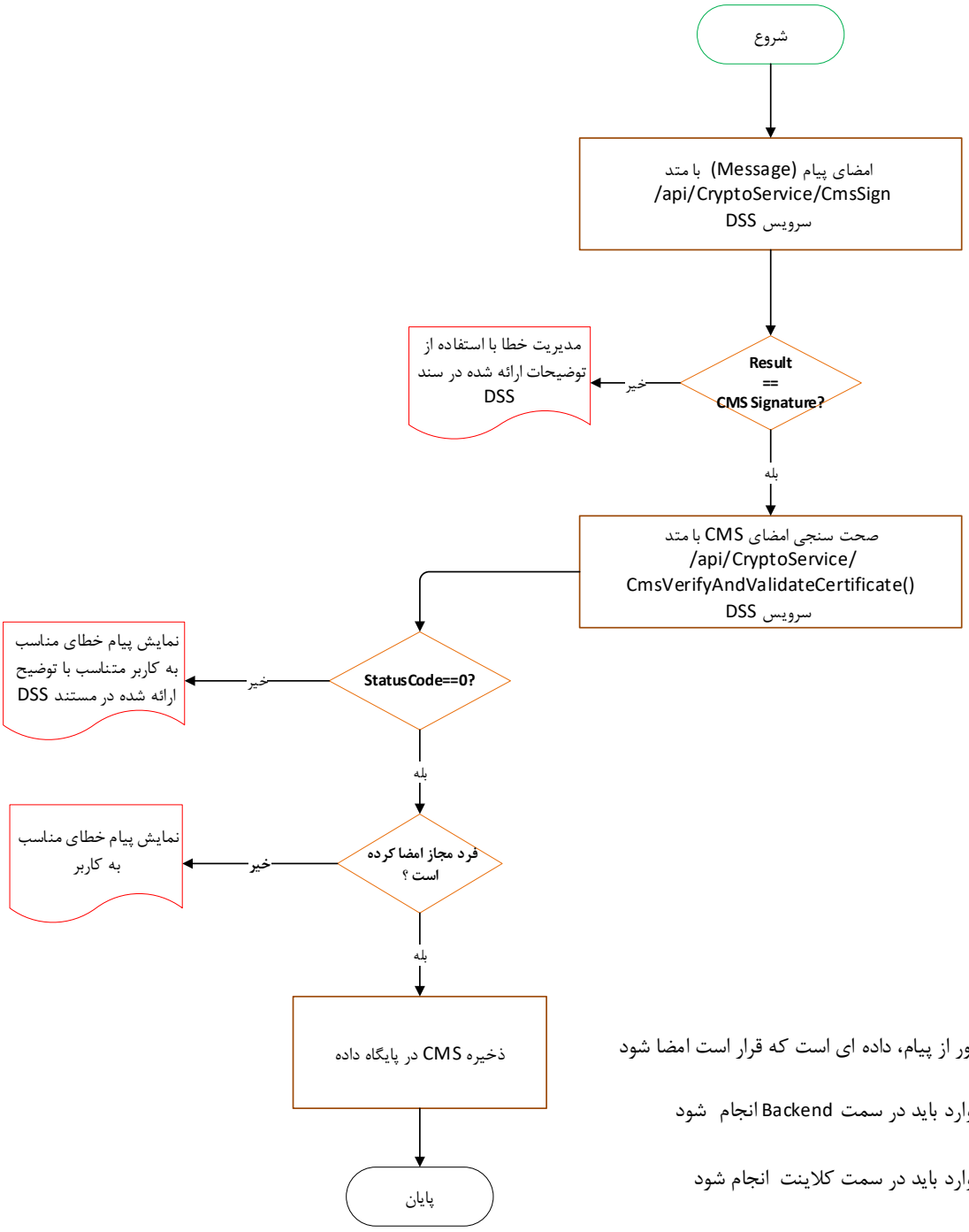
- اگر امضا از نوع CMSAttach میباشد از متد /api/CryptoService/CmsVerifyAndValidateCertificateAttach در سرویس DSS استفاده نمایید
- اگر امضا از نوع CMSDetach میباشد از متد /api/CryptoService/CmsVerifyAndValidateCertificate در سرویس DSS استفاده نمایید

۵/۹/۳ امضای **CMS** توسط توکن نرم افزاری

نمودار توالی امضای CMS توسط توکن نرم افزاری



فلوچارت توالی امضای CMS توسط توکن نرم افزاری



پیام: منظور از پیام، داده ای است که قرار است امضا شود

این موارد باید در سمت Backend انجام شود

این موارد باید در سمت کلاینت انجام شود

فهرست توابع موردنیاز جهت امضای CMS با توکن نرم افزاری:

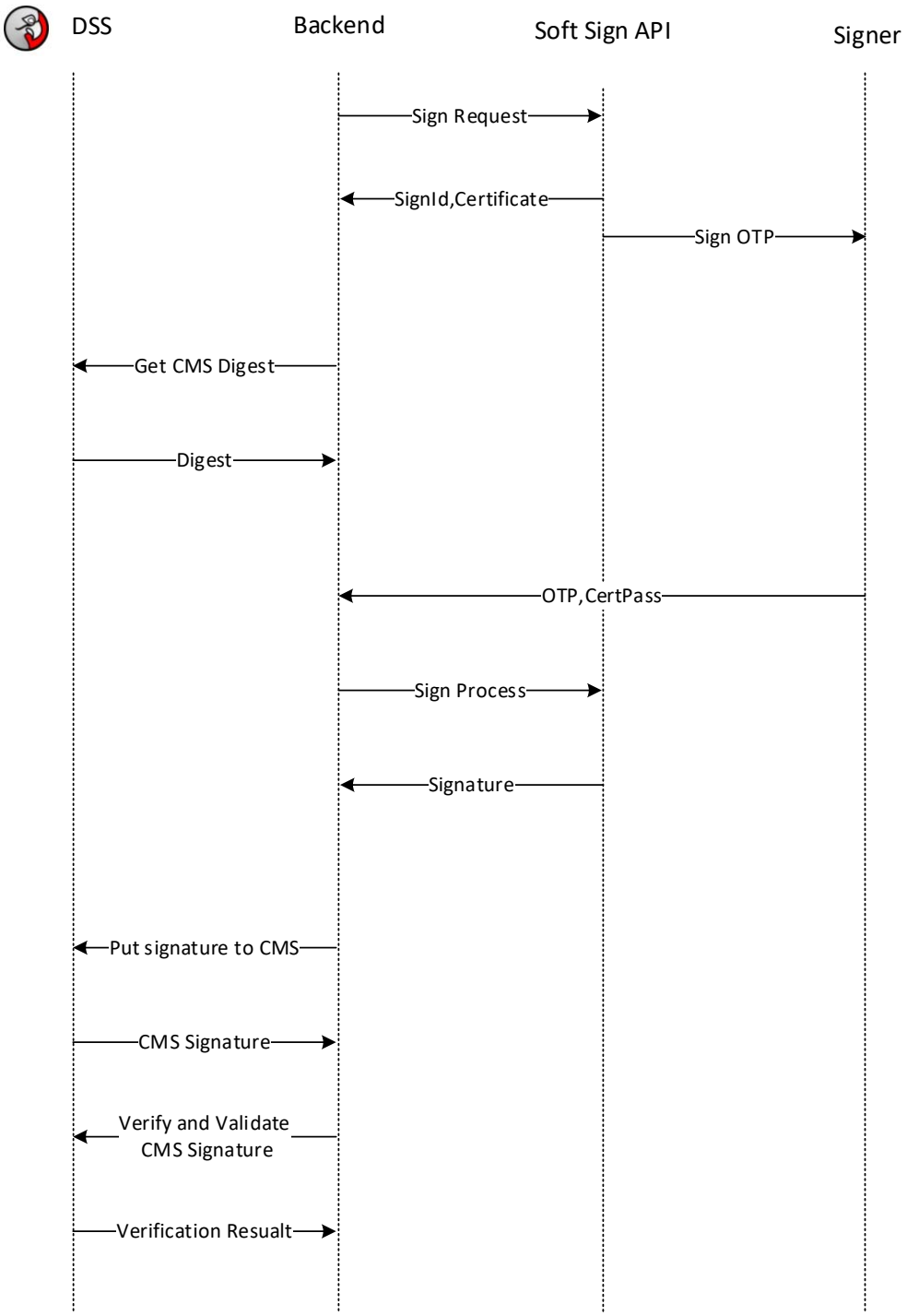
۱. امضای پیام با فراخوانی متد /api/CryptoService/CMSSign در سرویس DSS

۲. صحت سنجی امضای CMS با فراخوانی یکی از متدهای زیر در سرویس DSS

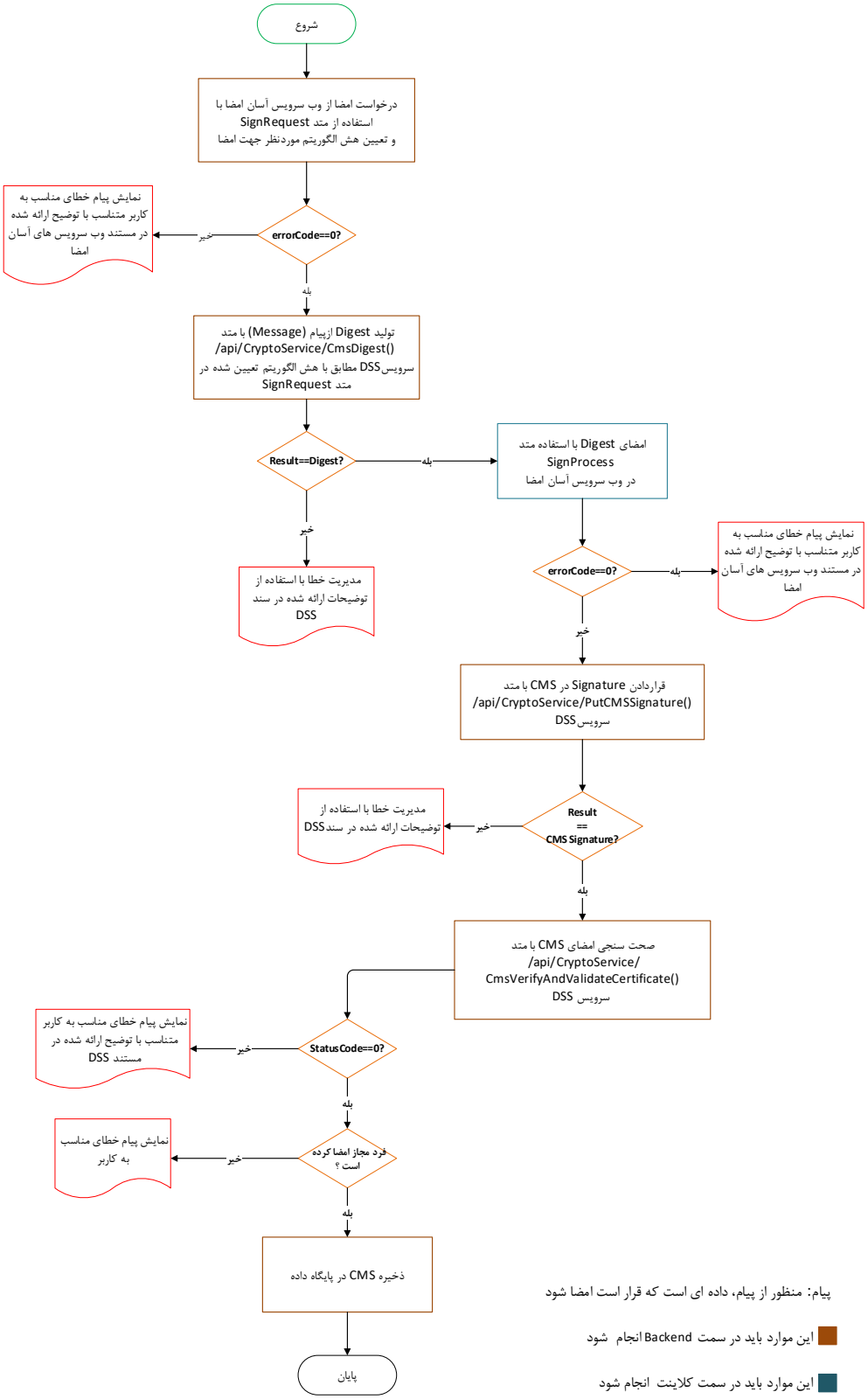
- درصورتی که امضا از نوع CMSAttach میباشد از متد
/api/CryptoService/CmsVerifyAndValidateCertificateAttach در سرویس DSS استفاده نمایید
- درصورتی که امضا از نوع CMSDetach میباشد از متد
/api/CryptoService/CmsVerifyAndValidateCertificate در سرویس DSS استفاده نمایید

۵/۹/۴ امضای **CMS** توسط ابر آسان امضا

نمودار توالی امضای CMS توسط ابر آسان امضا



فلوچارت توالی امضای CMS توسط ابر آسان امضا



فهرست توابع موردنیاز جهت امضای CMS توسط ابر آسان امضا:

۱. درخواست امضا با استفاده از عمل SignRequest در وب سرویس های آسان امضا
۲. تولید Digest از پیام با استفاده از متد /api/CryptoService/CmsDigest در سرویس DSS مطابق با هش الگوریتم تعیین شده حین استفاده از عمل SignRequest
۳. امضای Digest تولید شده با استفاده از عمل SignProcess در وب سرویس های آسان امضا با پارامترهای ورودی زیر:
   - signId : شناسه امضا (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقداردهی شود)
   - dataforsign : Digest تولید شده با فرمت base64
   - password : رمز گواهی (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقداردهی شود)
   - otp : رمز امضا (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقدادهی شود)
   - pkcs1support : True
٤. قراردادن امضا در CMS با فراخوانی متد /api/CryptoService/PutCMSSignature در سرویس DSS
٥. صحت سنجی امضای CMS با فراخوانی یکی از متدهای زیر در سرویس DSS
   - درصورتی که امضا از نوع CMSAttach میباشد از متد /api/CryptoService/CmsVerifyAndValidateCertificateAttach در سرویس DSS استفاده نمایید
   - درصورتی که امضا از نوع CMSDetach میباشد از متد /api/CryptoService/CmsVerifyAndValidateCertificate در سرویس DSS استفاده نمایید

api/CryptoService/CmsSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CMSSignByCertificateRequest<string>
	{
	"message": "string",
	پیام موردنظر جهت امضا با فرمت Base64
	"certificate": "string",
	فایل p12 یا pfx (حاوی گواهی و کلید خصوص) با فرمت Base64
	"attachData": bool,
	مقدار boolean جهت انتخاب قراردادن متن پیام در قالب CMS
	"hashAlgorithm": HashAlgorithm جدول شماره ۴
	الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضا
	"certificatePassword": "string" توجه
	رمز فایل p12 یا pfx
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	امضای پیام با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

درصورتی که فایل p12 یا pfx فاقد رمز باشد باید مقدار certificatePassword برابر با null درنظرگرفته شود.

api/CryptoService/CmsDigest	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0
Body	CmsDigestRequest<string> { "message": "string" ,

	متن پیام با فرمت Base64 "hashAlgorithm": HashAlgorithm جدول شماره ۴ الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضای پیام "certificate": "string", گواهی امضاکننده با فرمت Base64 "signDate": "dateTime" توجه زمان امضا }
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	Digest پیام با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

- زمان امضا را باید برابر با زمان کنونی قرار دهید.

api/CryptoService/PutCMSSignature	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0
Body	CMSSignature<string> { "message": "string", متن پیام با فرمت Base64 "hashAlgorithm": HashAlgorithm, توجه ۱ الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضای پیام "certificate": "string", گواهی امضاکننده با فرمت Base64 "signDate": "dateTime" , توجه ۲ زمان امضا "signature": "string", توجه ۳ امضا Digest پیام با فرمت Base64 "encapsulate": bool مقدار boolean جهت انتخاب قراردادن متن پیام در قالب CMS

	}
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	امضا پیام با فرمت Base64 در قالب CMS

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. مقدار HashAlgorithm باید برابر با مقدار درنظر گرفته شده در تابع CmsDigest باشد. برای اطلاع از مقادیر پارامتر hashAlgorithm به جدول شماره ۴ مراجعه نمایید.

۲. توجه داشته باشید که مقدار زمان امضا در متد PutCMSSignature زمان کنونی **نیست** و برابر با مقدار زمان امضا در تابع CmsDigest می باشد.

۳. Digest بدست آمده از خروجی متد CmsDigest، ابتدا باید از طریق گواهی صاحب امضا، مطابق با شرایط زیر امضا شود و سپس به عنوان مقدار پارامتر signature در متد PutCMSSignature قرارگیرد

- Digest سند PDF باید مطابق با استاندارد PKCS#1 امضا شود
- الگوریتم درهم‌سازی مورد استفاده در عملیات امضا، باید مطابق با مقدار درنظرگرفته شده برای پارامتر HashAlgorithm باشد

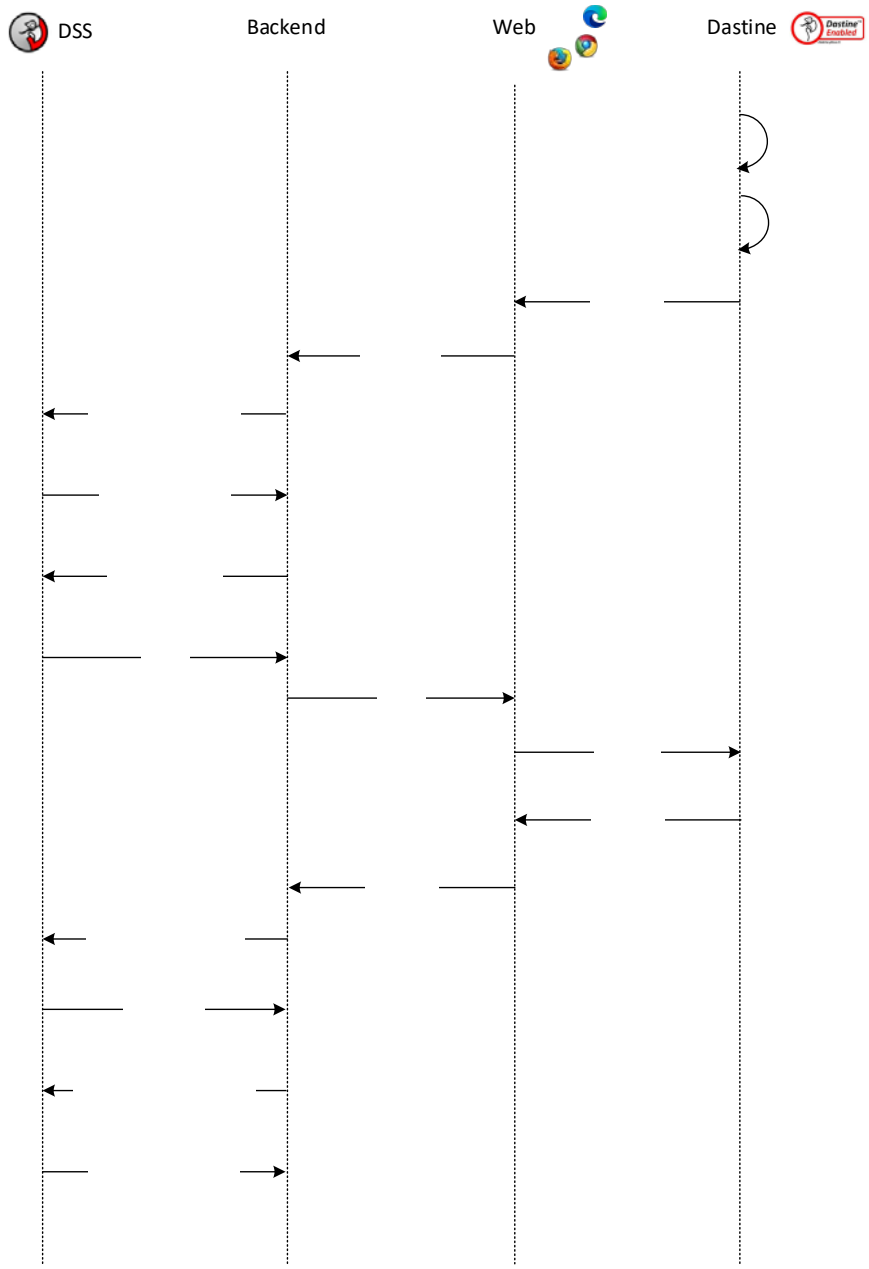
۴. **تمامی پارامترهای مشترک در دو متد CmsDigest و PutCMSSignature باید دارای مقادیر یکسان باشند.**

۵/۱۰امضای چندگانه سند PDF

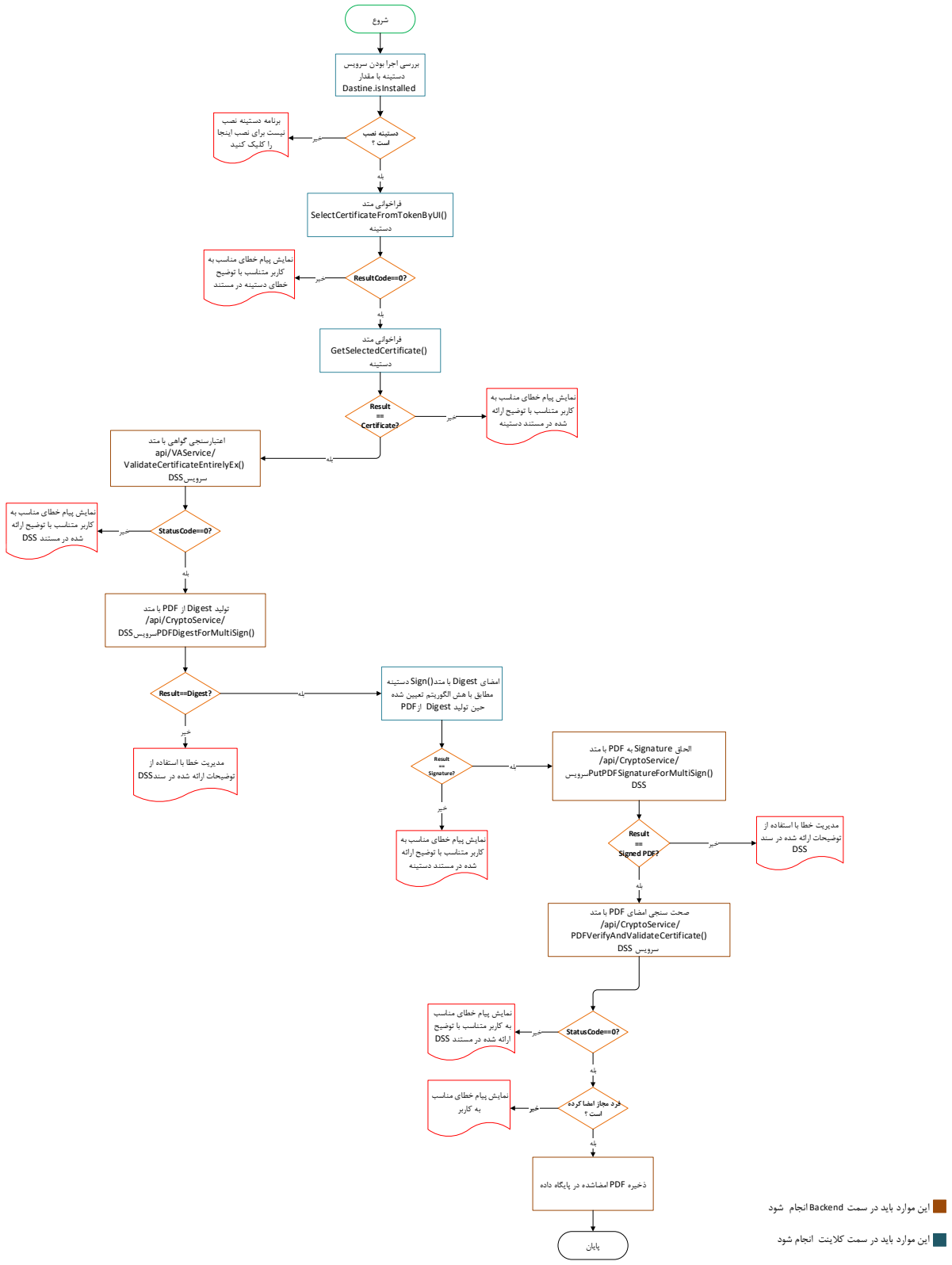
۵/۱۰/۱امضای **PDF** توسط دستینه

الزامیست نسخه PDF ۱.۶ یا بالاتر باشد

نمودار توالی امضایPDF توسط دستینه



فلوچارت توالی امضایPDF توسط دستینه



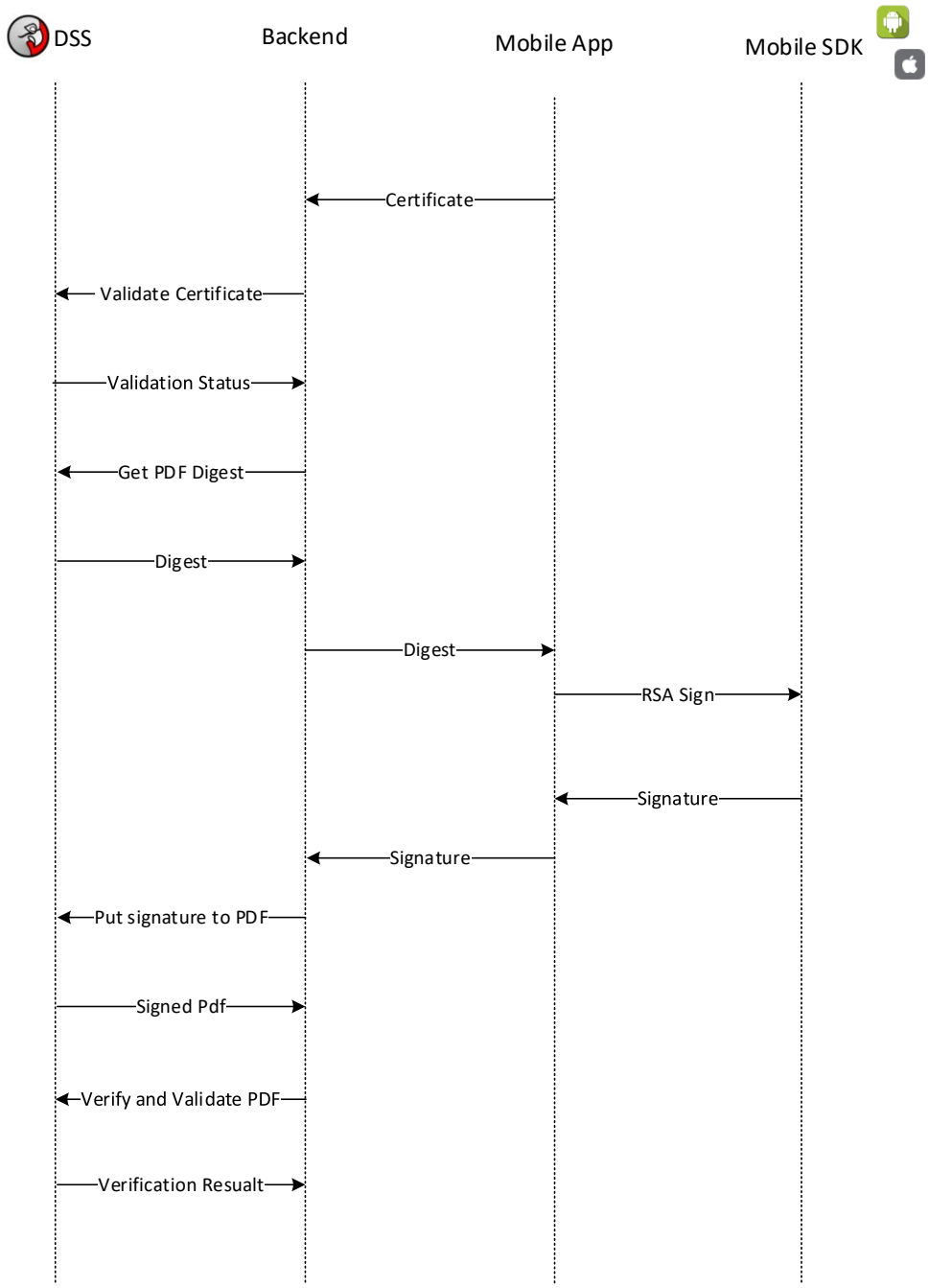
فهرست توابع موردنیاز جهت امضای PDF با دستینه :

۱. انتخاب گواهی موردنظر جهت امضا با فراخوانی متد SelectCertificateFromTokenByUI() در دستینه
۲. دریافت گواهی انتخاب شده با فرمت Base64 توسط متد GetSelectedCertificate() در دستینه
۳. اعتبارسنجی گواهی از طریق متد /api/VAService/ValidateCertificateEntirelyEx در سرویس DSS
٤. تولید Digest از PDF موردنظر با استفاده از متد /api/CryptoService/PDFDigestForMultiSign در سرویس DSS
۵. استفاده از متد Sign() دستینه با پارامترهای ورودی زیر جهت امضای Digest تولید شده
   - string data : Didgest تولید شده با فرمت Base64
   - String hashAlg : الگوریتم هش استفاده شده حین تولید Digest از PDF
٦. الحاق امضا به PDF با فراخوانی متد /api/CryptoService/PutPDFSignatureForMultiSign در سرویس DSS
۷. صحت سنجی امضای PDF با فراخوانی متد /api/CryptoService/PDFVerifyAndValidateCertificate در سرویس DSS

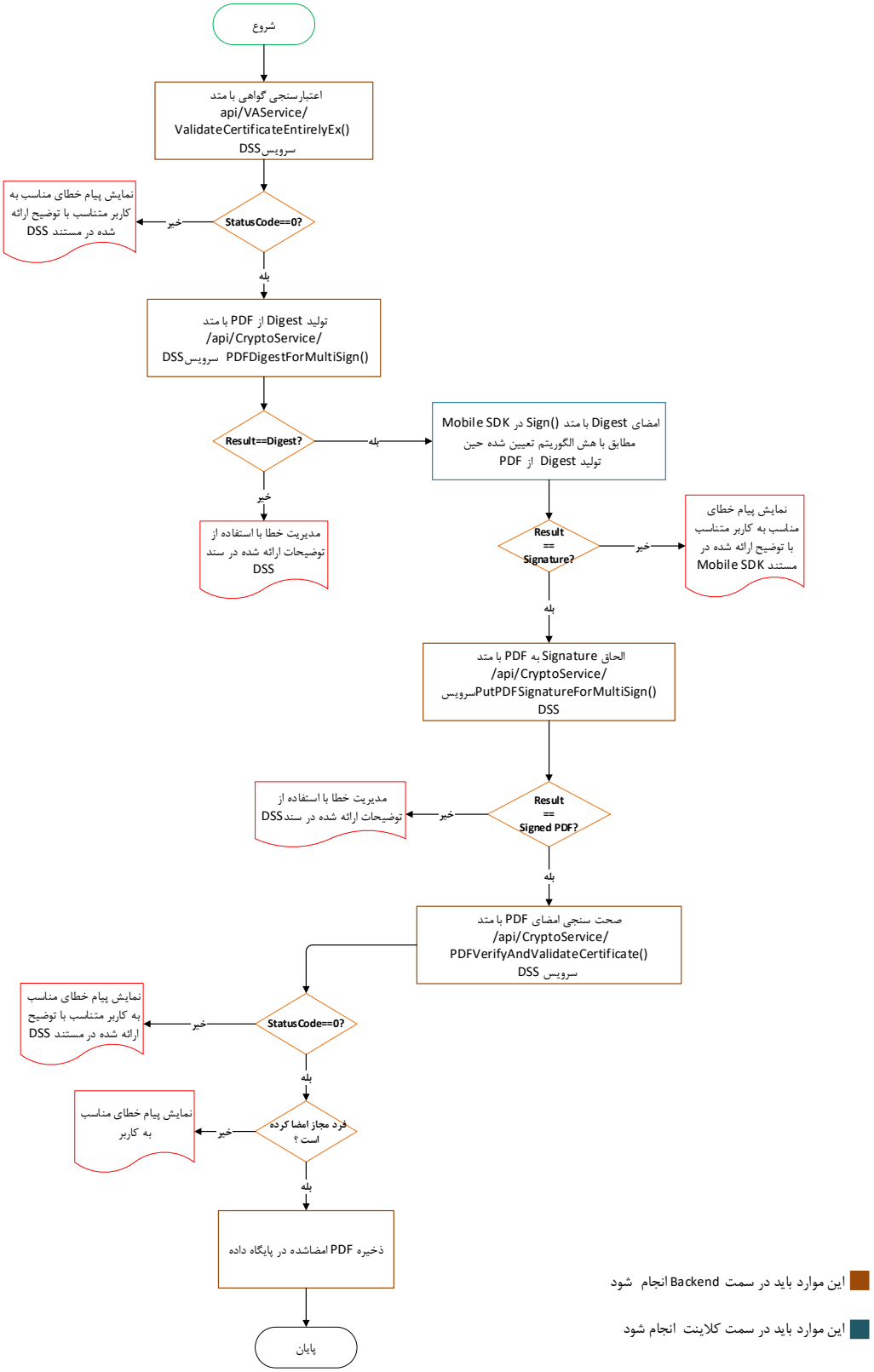
۵/۱۰/۲امضای **PDF** توسط **SDK** موبایل

الزامیست نسخه PDF ۱.۶ یا بالاتر باشد

نمودار توالی امضایPDF توسط SDK موبایل



فلوچارت توالی امضایPDF توسط SDK موبایل



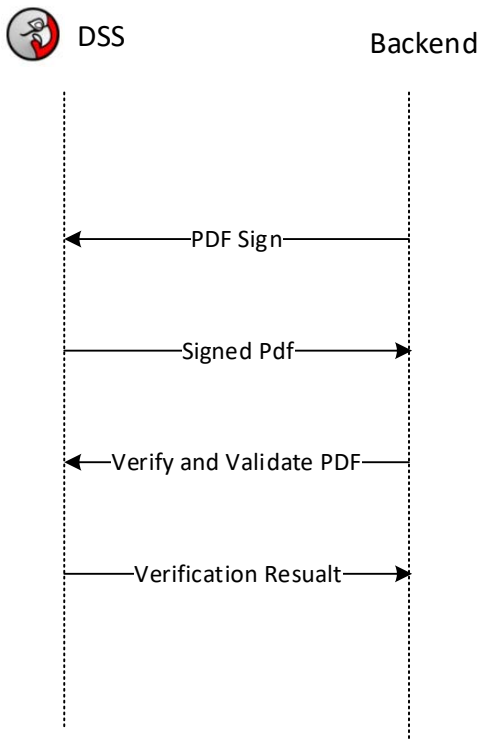
فهرست توابع موردنیاز جهت امضای PDF توسط SDK موبایل :

۱. اعتبارسنجی گواهی امضا از طریق متد /api/VAService/ValidateCertificateEntirelyEx در سرویس DSS
۲. تولید Digest از PDF موردنظر با استفاده از متد /api/CryptoService/PDFDigestForMultiSign در سرویس DSS
۳. استفاده از متد ()Sign در Mobile SDK با پارامترهای ورودی زیر جهت امضای Digest تولید شده :
   - string message : Didgest تولید شده با فرمت Base64
   - string privateKeyName : نام کلید خصوصی (مطابق با توضیحات ارائه شده در سند SDK موبایل مقداردهی شود)
   - string hashAlg : الگوریتم هش استفاده شده حین تولید Digest از PDF
   - string pin : پین کلید خصوصی (مطابق با توضیحات ارائه شده در سند SDK موبایل مقداردهی شود)
٤. الحاق امضا به PDF با فراخوانی متد /api/CryptoService/PutPDFSignatureForMultiSign در سرویس DSS
٥. صحت سنجی امضای PDF با فراخوانی متد /api/CryptoService/PDFVerifyAndValidateCertificate در سرویس DSS

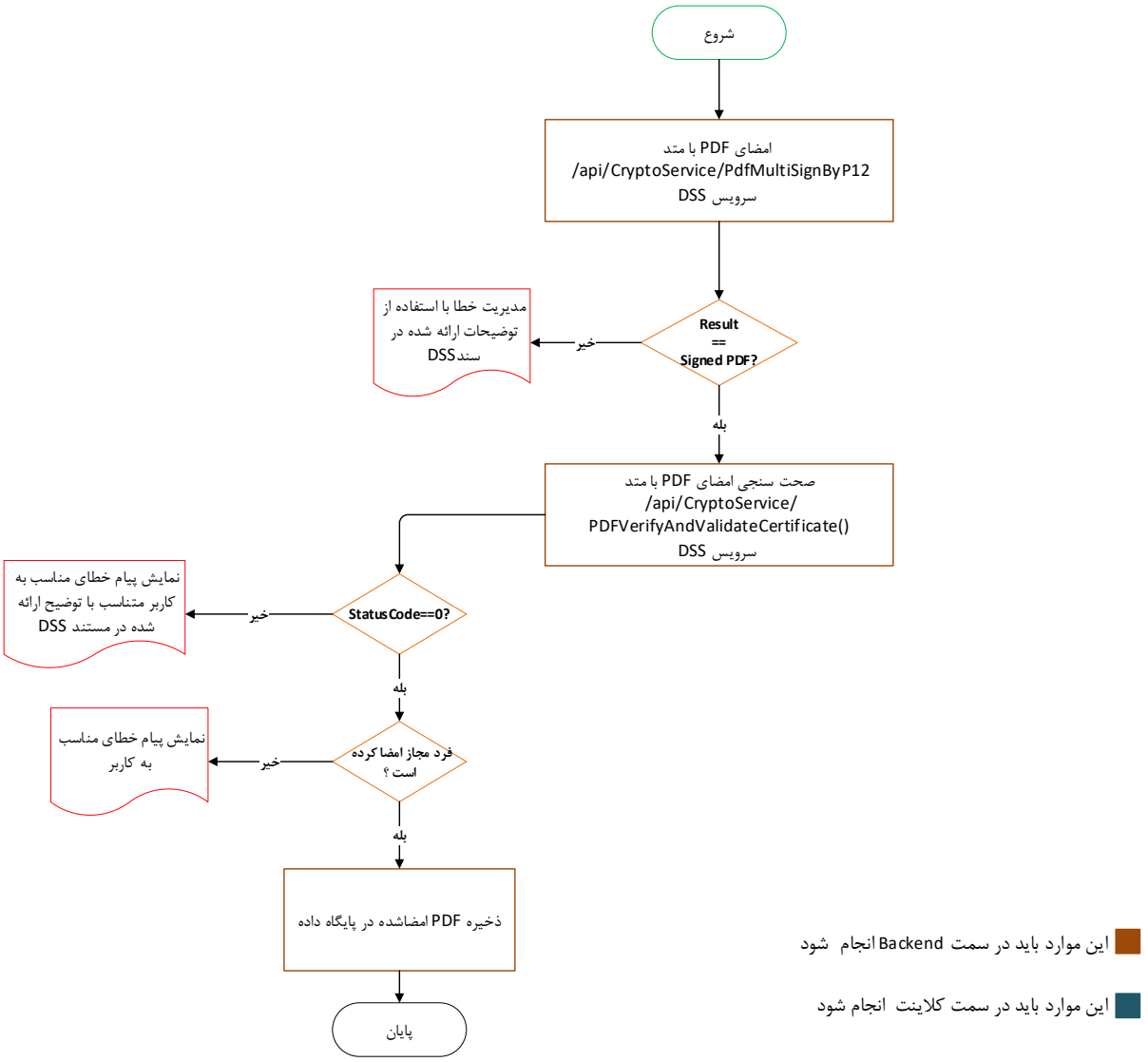
۵/۱۰/۳ امضای **PDF** توسط توکن نرم افزاری

الزامیست نسخه PDF ۱.۶ یا بالاتر باشد

نمودار توالی امضای PDF توسط توکن نرم افزاری



فلوچارت توالی امضایPDF توسط توکن نرم افزاری



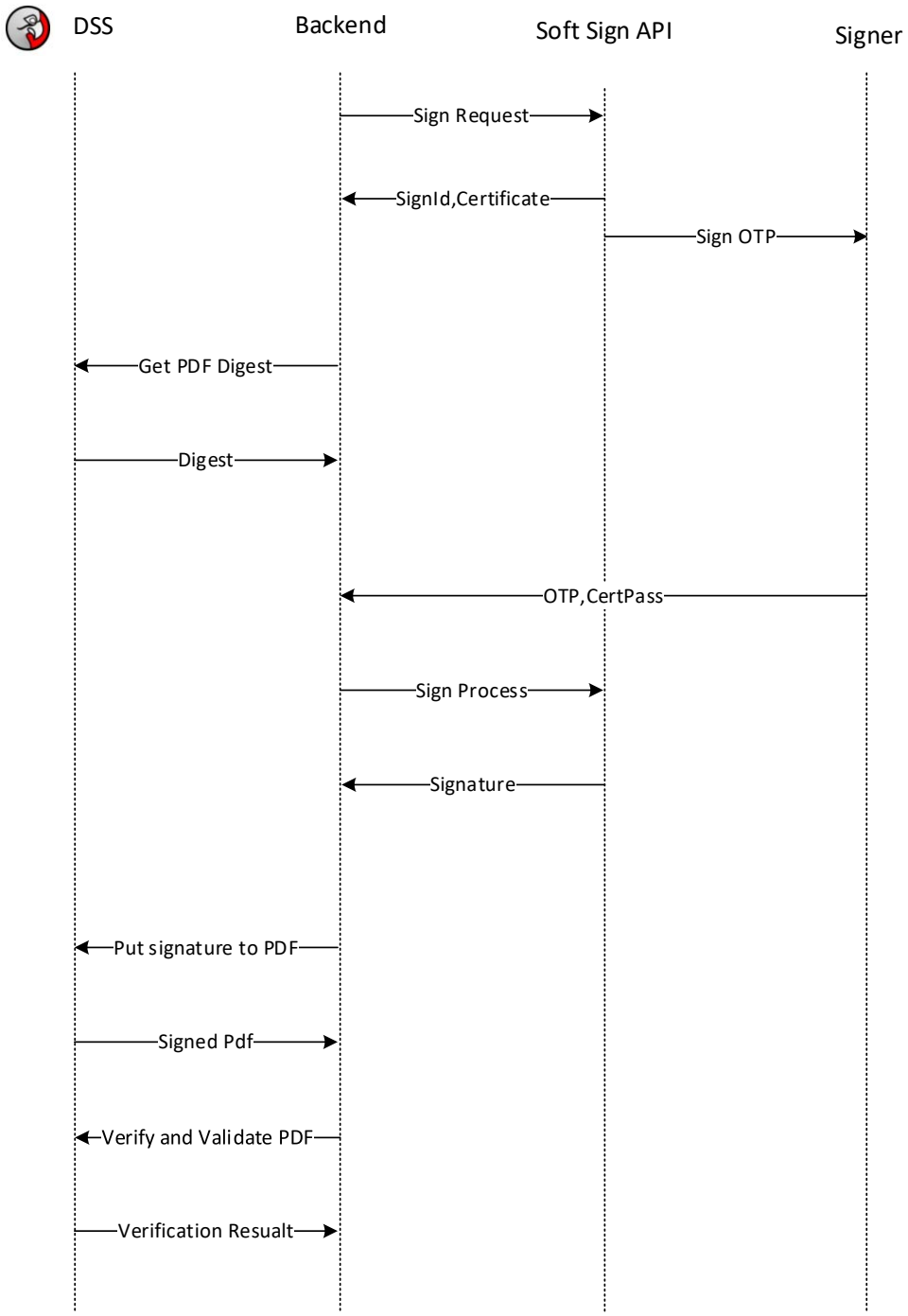
فهرست توابع موردنیاز جهت امضای PDF با توکن نرم افزاری :

۱. امضای PDF با فراخوانی متد /api/CryptoService/PdfMultiSignByP12 در سرویس DSS
۲. صحت سنجی امضای PDF با فراخوانی متد /api/CryptoService/PDFVerifyAndValidateCertificate در سرویس DSS

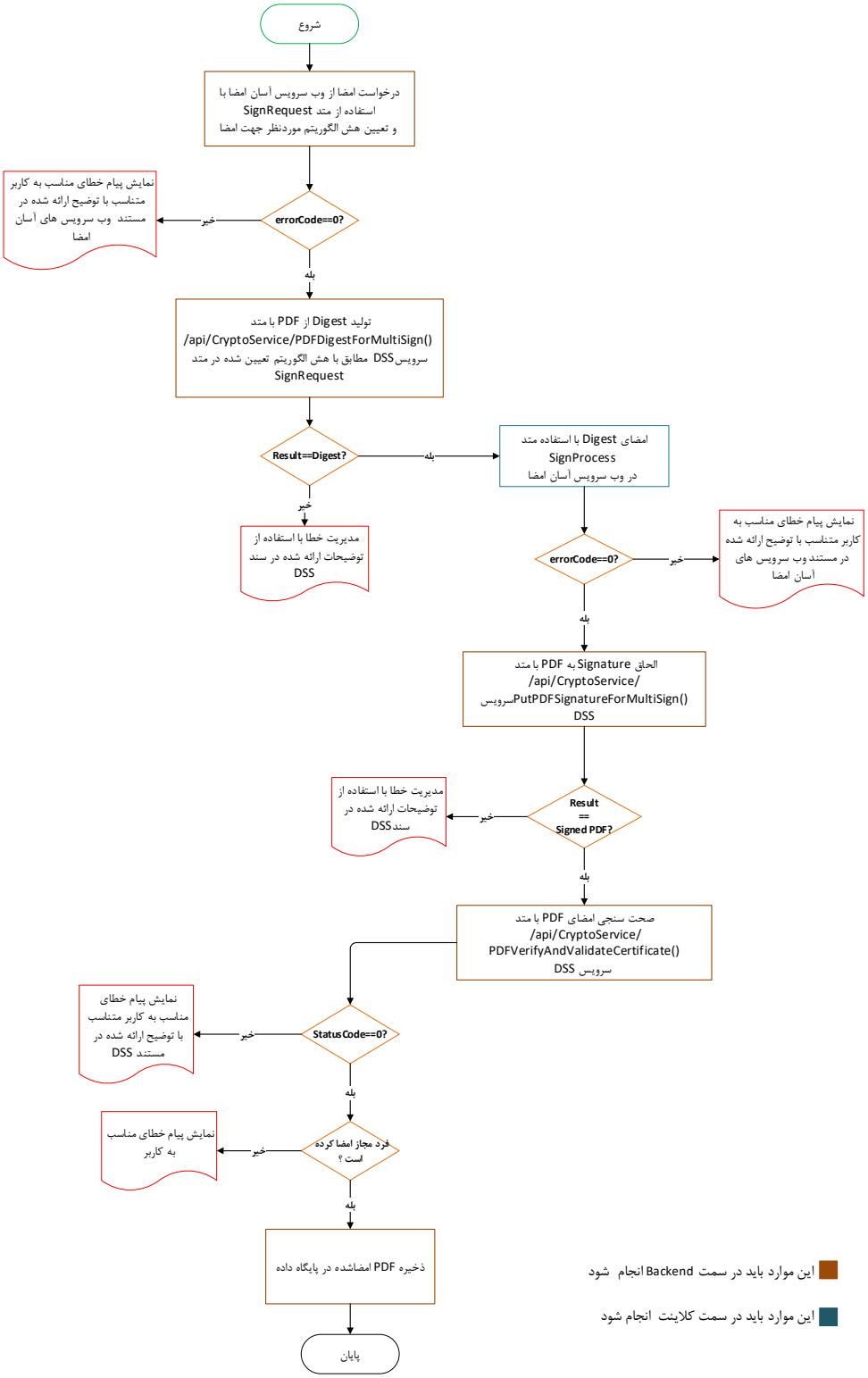
۵/۱۰/۴امضای **PDF** توسط ابر آسان امضا

الزامیست نسخه PDF ۱.۶ یا بالاتر باشد

نمودار توالی امضایPDF توسط ابر آسان امضا



فلوچارت توالی امضایPDF توسط ابر آسان امضا



فهرست توابع موردنیاز جهت امضای PDF توسط ابر آسان امضا:

۱. درخواست امضا با استفاده از عمل SignRequest در وب سرویس های آسان امضا
۲. تولید Digest از PDF موردنظر با استفاده از متد /api/CryptoService/PDFDigestForMultiSign در سرویس DSS مطابق با هش الگوریتم تعیین شده حین استفاده از عمل SignRequest
۳. امضای Digest تولید شده با استفاده از عمل SignProcess در وب سرویس های آسان امضا با پارامترهای ورودی زیر:
   - signId : شناسه امضا (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقدادهی شود)
   - dataforsign : Digest تولید شده با فرمت base64
   - password : رمز گواهی (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقدادهی شود)
   - otp : رمز امضا (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقدادهی شود)
   - pkcs1support : True
٤. الحاق امضا به PDF با فراخوانی متد /api/CryptoService/PutPDFSignatureForMultiSign در سرویس DSS
٥. صحت سنجی امضای PDF با فراخوانی متد /api/CryptoService/PDFVerifyAndValidateCertificate در سرویس DSS

api/CryptoService/PDFDigestForMultiSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	MultiSignPdfDigestRequest<string>
	{
	"pdfData": "string",
	سند PDF با فرمت Base64
	"signerCertificate": "string",
	گواهی امضاکننده با فرمت Base64
	"certificationLevel": ۰, توجه ۱
	سطح دسترسی امضا
	"dateTime": "dateTime", توجه ۵
	زمان امضا
	"reason": "string", اختیاری
	علت امضا
	"location": "string", اختیاری
	نام موقعیت مکانی
	"imageDataUrl": "string", اختیاری
	تصویر موردنظر جهت درج در سند امضاشده با فرمت Base64
	"page": number, توجه ۳
	شماره صفحه سند جهت درج تصویر امضا
	"lowerLeftX": number, توجه ۲
	موقعیت تصویر در صفحه
	"lowerLeftY": number, توجه ۲
	موقعیت تصویر در صفحه
	"upperRightX": number, توجه ۲
	موقعیت تصویر در صفحه
	"upperRightY": number, توجه ۲
	موقعیت تصویر در صفحه
	"signatureFieldName": "string", توجه ۴
	نام فیلد امضا
	"hashAlgorithm": HashAlgorithm جدول شماره ۴
	الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضای سند
	}

Response	

Status Code	200 Success / 400 BadRequest
200 Schema	Digest سند PDF با فرمت Base64

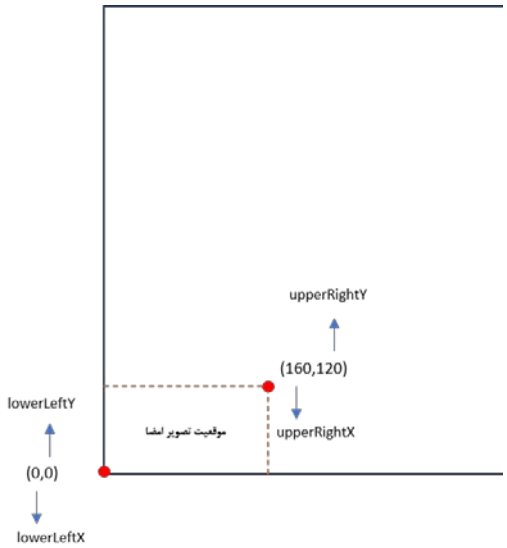
	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. مادامی که نیازاست سند مجددا امضا شود سطح دسترسی امضا باید برابر با "۰" قرارگیرد. برای اطلاع از سایر مقادیر پارامتر certificationLevel به جدول شماره ۹ مراجعه نمایید.

۲. موقعیت تصویر براساس X+ و Y+ در محور مختصات محاسبه می‌گردد. به بیان دیگر نقطه (0, 0) در سمت چپ و پایین صفحه PDF قرار دارد.

به طور مثال :

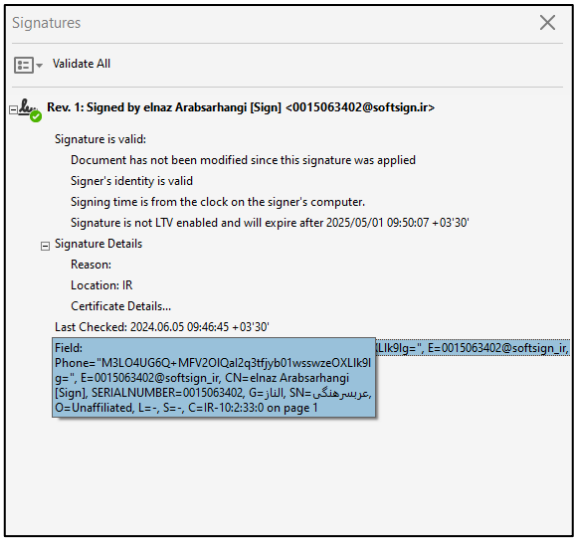


توجه داشته باشید که تعیین مختصات تصویر در شرایطی نیاز است که پارامتر imageDataUrl مقداردهی شده باشد به بیانی دیگر درصورت عدم تمایل به درج تصویر امضا نیازی به تعیین مختصات تصویر نمی‌باشد.

۳. مقدار پارامتر page برابر است با شماره صفحه موردنظر جهت درج تصویر امضا در سند ولی در شرایطی که نیازی به درج تصویر امضا در سند نمی‌باشد مقدار این پارامتر باید برابر با ۱ درنظر گرفته شود

٤. فیلد امضا (signatureFieldName) شامل یک نام متمایز برای هر امضا می باشد که باید توسط توسعه دهنده، مقدار دهی شود. برای مشاهده فیلد امضا، کافی است سند امضا شده را در Adobe Reader اجرا نمایید تا در پنل سمت چپ، نام فیلد نمایش داده شود

در تصویر نمونه زیر, اقلام گواهی (SubjectDN) بعنوان نام فیلد امضا درنظر گرفته شده است.



٥. زمان امضا را باید برابر با زمان کنونی قرار دهید.

api/CryptoService/PutPDFSignatureForMultiSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0
Schema	MultiSignPdfSignature<string> { "pdfData": "string", سند PDF با فرمت Base64 "signerCertificate": "string", گواهی امضاکننده، با فرمت Base64 "certificationLevel": ۰, توجه ۱ سطح دسترسی امضا "dateTime": "dateTime", توجه ۷ زمان امضا "reason": "string", اختیاری علت امضا "location": "string", اختیاری نام موقعیت مکانی "imageDataUrl": "string", اختیاری تصویر موردنظر جهت درج در سند امضاشده با فرمت Base64 "page": number, توجه ۳ شماره صفحه سند جهت درج تصویر امضا

	"lowerLeftX": number, توجه ۲ موقعیت تصویر در صفحه "lowerLeftY": number, توجه ۲ موقعیت تصویر در صفحه "upperRightX": number, توجه ۲ موقعیت تصویر در صفحه "upperRightY": number, توجه ۲ موقعیت تصویر در صفحه "signatureFieldName": "string", توجه ۴ نام فیلد امضا "hashAlgorithm": HashAlgorithm, توجه ۵ الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضای سند "signature": "string" توجه ۶ Digest امضاشده از سند PDF با فرمت Base64 }
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	سند PDF امضا شده با فرمت Base64

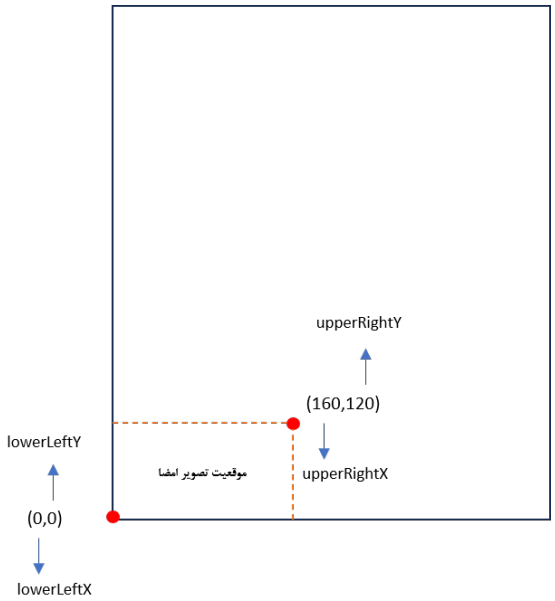
	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه :

۱. مقدار certificationLevel باید برابر با مقدار درنظر گرفته شده در تابع PDFDigestForMultiSign باشد. برای اطلاع از مقادیر پارامتر certificationLevel به جدول شماره ۹ مراجعه نمایید.

۲. موقعیت تصویر براساس X+ و Y+ در محور مختصات محاسبه می‌گردد. به بیان دیگر نقطه (0, 0) در سمت چپ و پایین صفحه PDF قرار دارد.

به طور مثال :



توجه داشته باشید که تعیین مختصات تصویر در شرایطی نیاز است که پارامتر imageDataUrl مقداردهی شده باشد به بیانی دیگر درصورت عدم تمایل به درج تصویر امضا نیازی به تعیین مختصات تصویر نمی‌باشد.

۳. مقدار پارامتر page برابر است با شماره صفحه موردنظر جهت درج تصویر امضا در سند ولی در شرایطی که نیازی به درج تصویر امضا در سند نمی‌باشد مقدار این پارامتر باید برابر با ۱ درنظر گرفته شود

۴. فیلد امضا (signatureFieldName) شامل یک نام متمایز برای هر امضا می باشد که باید توسط توسعه دهنده مقدار دهی شود. برای مشاهده فیلد امضا، کافی است سند امضا شده را در Adobe Reader اجرا نمایید تا در پنل سمت چپ، مجموعه اطلاعات امضا براساس نام فیلد نمایش داده شود

۵. مقدار hashAlgorithm باید برابر با مقدار درنظر گرفته شده در تابع PDFDigestForMultiSign باشد. برای اطلاع از مقادیر پارامتر hashAlgorithm به جدول شماره ۴ مراجعه نمایید.

۶. Digest بدست آمده از خروجی متد PDFDigestForMultiSign، ابتدا باید از طریق گواهی صاحب امضا، مطابق با شرایط زیر امضا شود و سپس به عنوان مقدار پارامتر signature در متد PutPDFSignatureForMultiSign قرارگیرد

- Digest سند PDF باید مطابق با استاندارد PKCS#1 امضا شود
- الگوریتم درهم‌سازی مورد استفاده در عملیات امضا، باید مطابق با مقدار درنظرگرفته شده برای پارامتر hashAlgorithm باشد

۷. توجه داشته باشید که مقدار زمان امضا در متد PutPDFSignatureForMultiSign زمان کنونی نیست و مقدار قبلی در تابع PDFDigestForMultiSign می باشد.

۸. تمامی پارامترهای مشترک در دو متد PDFDigestForMultiSign و PutPDFSignatureForMultiSign باید دارای مقادیر یکسان باشند.

api/CryptoService/PdfMultiSignByP12	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0
Schema	PdfMultiSignByP12Request<string> { "pdfData": "string", سند PDF با فرمت Base64 "signerCertificate": "string", گواهی و کلیدخصوصی امضاکننده (فایل P12) با فرمت Base64 "certificationLevel": ۰, توجه ۱ سطح دسترسی امضا "dateTime": dateTime, توجه ۵ زمان امضا "reason": "string", اختیاری علت امضا "location": "string", اختیاری نام موقعیت مکانی "imageDataUrl": "string", اختیاری تصویر موردنظر جهت درج در سند امضاشده با فرمت Base64 "page": number, توجه ۳ شماره صفحه سند جهت درج تصویر امضا "lowerLeftX": number, توجه ۲ موقعیت تصویر در صفحه "lowerLeftY": number, توجه ۲ موقعیت تصویر در صفحه "upperRightX": number, توجه ۲ موقعیت تصویر در صفحه "upperRightY": number, توجه ۲ موقعیت تصویر در صفحه "signatureFieldName": "string", توجه ۴ نام فیلد امضا "hashAlgorithm": HashAlgorithm, جدول شماره ۴ الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضای سند "Password": "string" رمز فایل P12

	}
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	سند PDF امضاشده با فرمت Base64

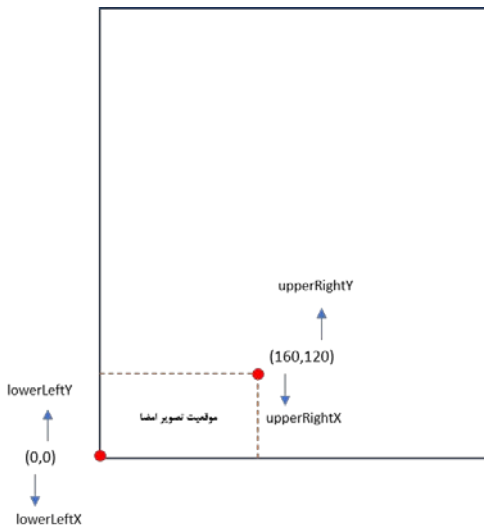
	{
	"error": "string",
	"result": "string",
	"statusCode": number
	}

توجه:

۱. مادامی که نیازاست سند مجددا امضا شود سطح دسترسی امضا باید برابر با "۰" قرارگیرد. برای اطلاع از سایر مقادیر پارامتر certificationLevel به جدول شماره ۹ مراجعه نمایید

۲. موقعیت تصویر براساس X+ و Y+ در محور مختصات محاسبه می‌گردد. به بیان دیگر نقطه (0, 0) در سمت چپ و پایین صفحه PDF قرار دارد.

به طور مثال :

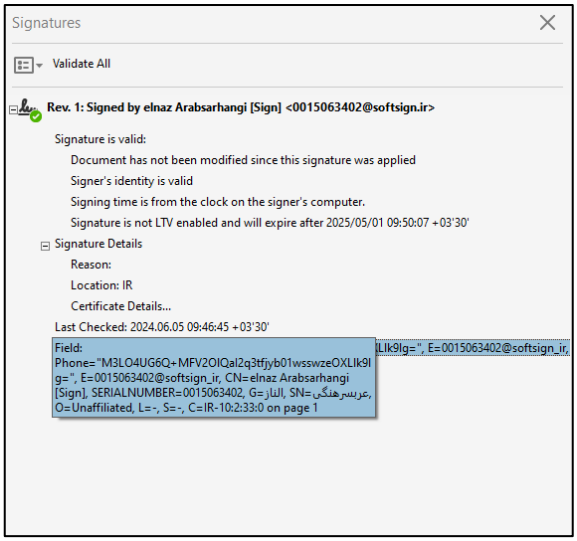


توجه داشته باشید که تعیین مختصات تصویر در شرایطی نیاز است که پارامتر imageDataUrl مقداردهی شده باشد به بیانی دیگر درصورت عدم تمایل به درج تصویر امضا نیازی به تعیین مختصات تصویر نمی‌باشد.

۳. مقدار پارامتر page برابر است با شماره صفحه موردنظر جهت درج تصویر امضا در سند ولی در شرایطی که نیازی به درج تصویر امضا در سند نمی‌باشد مقدار این پارامتر باید برابر با ۱ درنظر گرفته شود

٤. فیلد امضا (signatureFieldName) شامل یک نام متمایز برای هر امضا می باشد که باید توسط توسعه دهنده مقدار دهی شود. برای مشاهده فیلد امضا، کافی است سند امضا شده را در Adobe Reader اجرا نمایید تا در پنل سمت چپ، مجموعه اطلاعات امضا براساس نام فیلد نمایش داده شود

در تصویر نمونه زیر, اقلام گواهی (SubjectDN) بعنوان نام فیلد امضا درنظر گرفته شده است.



۵. زمان امضا را باید برابر با زمان کنونی قرار دهید.

۵/۱۱ استخراج گواهینامه‌ها از فایل PDF

api/CryptoService/PdfExtractCertificates	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	signedPdf : string سند PDF امضاشده با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از گواهی های استخراج شده با فرمت Base64

	{
	"error": "string",
	"result": IEnumerable<string>,
	"statusCode": number => 200 Success / 500 Error
	}

۵/۱۲ استخراج اطلاعات امضاها از فایل PDF

api/CryptoService/PDFExtractSignerInfo	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	signedPdf : string سند PDF امضاشده با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	از فهرستی از اطلاعات امضا

	{
	"error": "string",
	"result": IEnumerable<SignerInfo<string>>
	[
	"location": "string",
	نام موقعیت مکانی
	"reason": "string",
	علت امضا
	"revision": number,
	شمارنده امضا
	"signatureName": "string",
	نام فیلد امضا
	"signDate": "DateTime",
	زمان امضا
	"signingCertificate": "string",
	گواهی امضاکننده
	"certificationLevel": CertificationLevel جدول شماره ۹
	سطح دسترسی امضا
	],
	"statusCode": number => 200 Success / 500 Error
	}

۵/۱۳ تصدیق امضای دیجیتال PDF

api/CryptoService/PDFVerify	
Request	
Method	Post

Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Schema	signedPdf: string سند PDF امضاشده در قالبBase64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی امضای PDF

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۱۴ تصدیق امضای دیجیتال PDF و اعتبارسنجی گواهی امضا

api/CryptoService/PDFVerifyAndValidateCertificate	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	PdfVerifyAndValidateRequest<string>
	{
	"signedData": "string",
	سند PDF امضاشده، با فرمت Base64
	"vaProfile": "string" اختیاری
	نام پروفایل موردنظر جهت استفاده در عملیات اعتبارسنجی گواهی با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از نتایج بازگشتی از بررسی سند امضاشده
	{ "error": "string", "result": IEnumerable<VerificationResult> [{ "certificate": byte[], گواهی امضا با فرمت Base64

	"status": CMSVerifyValidationStatus, جدول شماره ۵ وضعیت امضا "revision": number, شمارنده ی امضا "result": bool نتیجه ارزیابی امضا }], "statusCode": number => 200 Success / 500 Error }

توجه:

در فرآیند ارزیابی سند امضاشده، مقداردهی پارمتر vaProfile اختیاری می‌باشد و ارسال آن توصیه نمی‌گردد. ولی درصورت نیاز به مقداردهی این پارامتر می‌توانید از تیم استقرار سرویس، راهنمایی های لازم را دریافت نمایید.

۵/۱۵استخراج گواهی از مهر زمانی

api/CryptoService/TstExtractCertificates	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	tst : string مهر زمانی صادر شده با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از گواهی های استخراج شده

	{
	"error": "string",
	"result": IEnumerable<string>,
	"statusCode": number => 200 Success / 500 Error
	}

۵/۱۶استخراج زمان از مهر زمانی

api/CryptoService/TstExtractTime	
Request	
Method	Post
Header	Content-Type: multipart/form-data

	Accept: text/plain api-version : 2.0 or 3.0
Body	tst : string مهر زمانی صادر شده با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	زمان استخراج‌شده

	{
	"error": "string",
	"result": "dateTime",
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۱۷ تصدیق مهر زمانی

api/CryptoService/TstVerify	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	TimestampVerifyRequest<string>
	{
	"message": "string",
	محتوای اولیه (قبل از امضا) با فرمت Base64
	"timestamp": "string",
	مهرزمانی صادر شده با فرمت Base64
	"certificate": "string"
	گواهی امضاکننده با فرمت Base64
	}

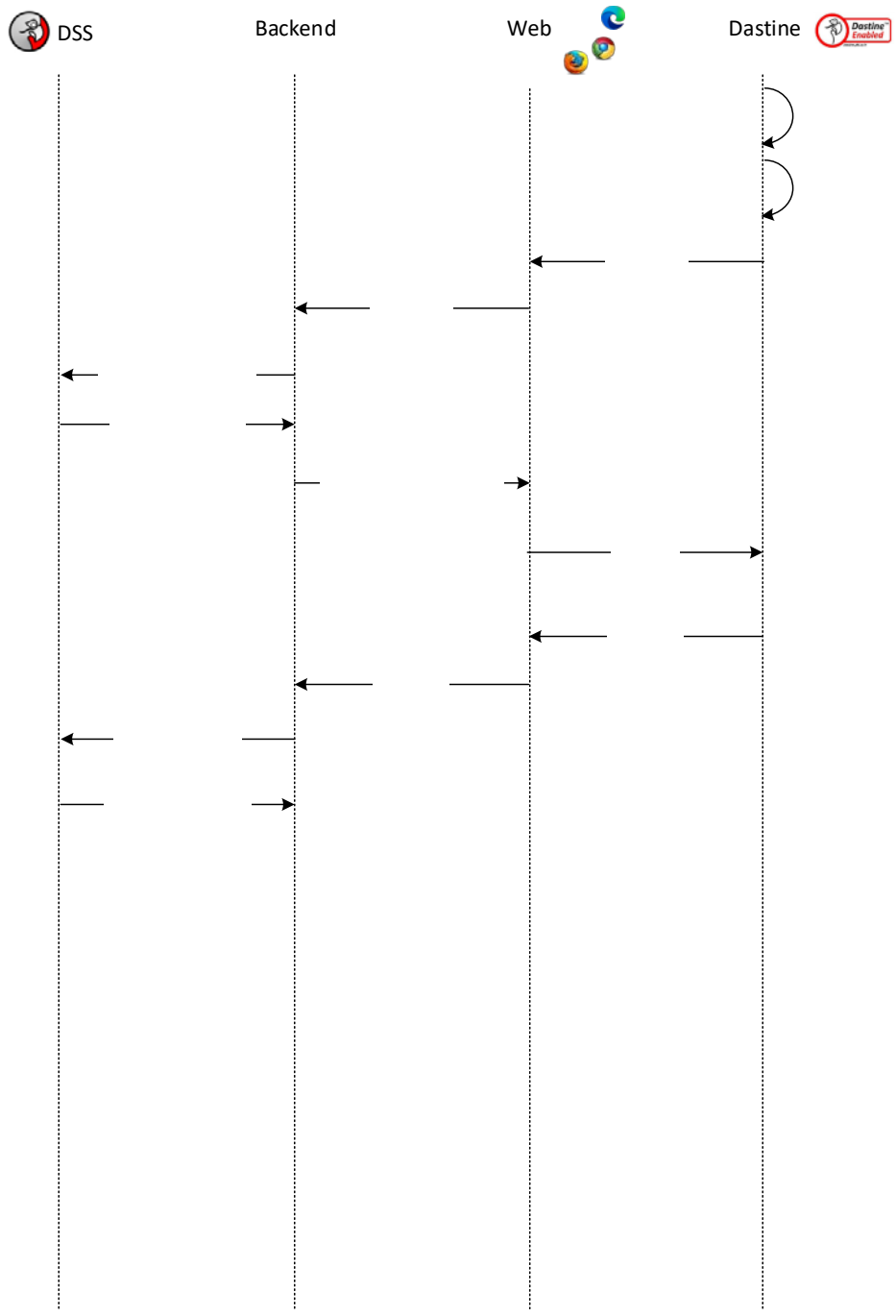
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی مهرزمانی

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

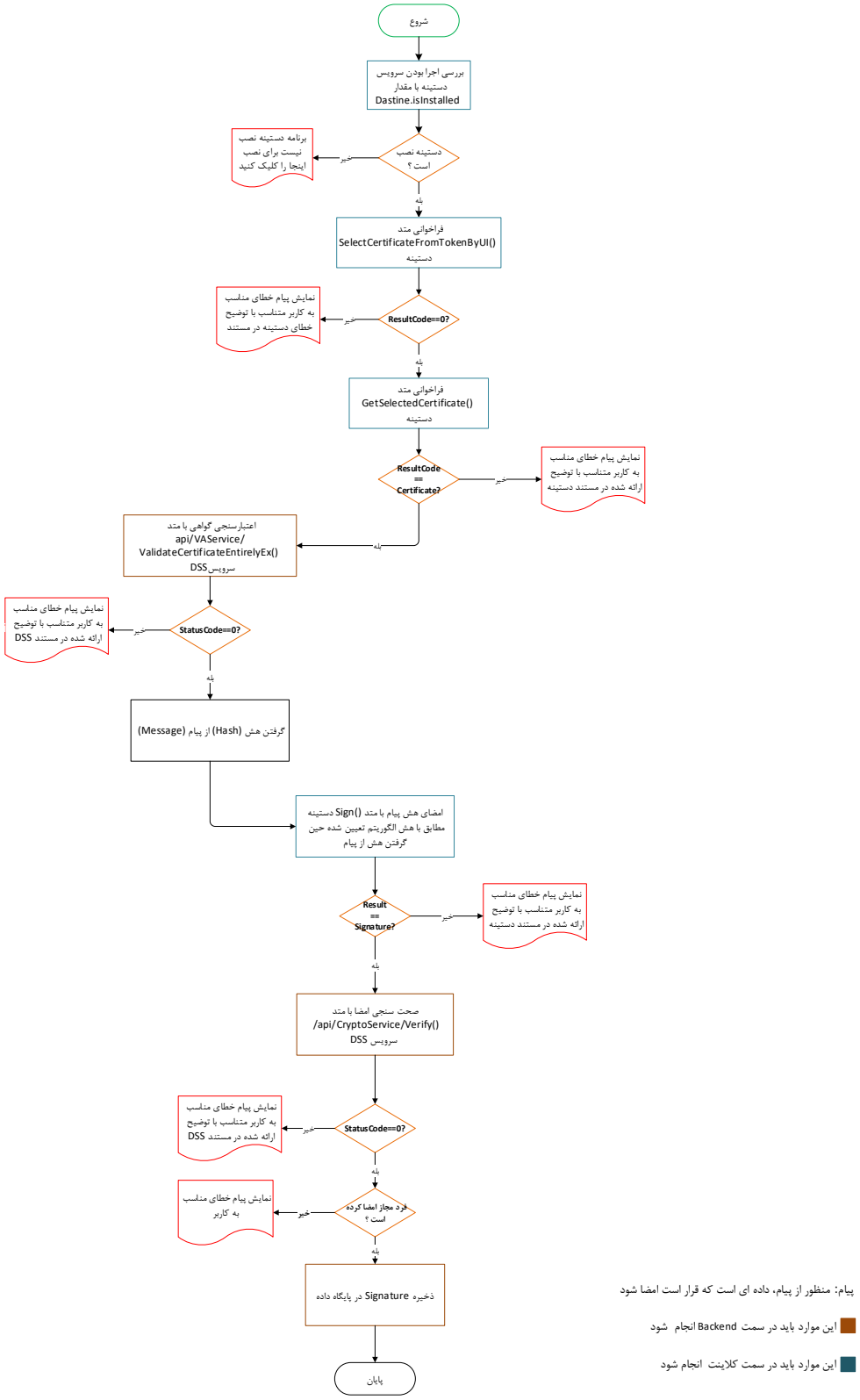
۵/۱۸امضای پیام براساس الگوریتم RSA

۵/۱۸/۱امضای **RSA** توسط دستینه

نمودار توالی امضای RSA توسط دستینه



فلوچارت توالی امضای RSA توسط دستینه

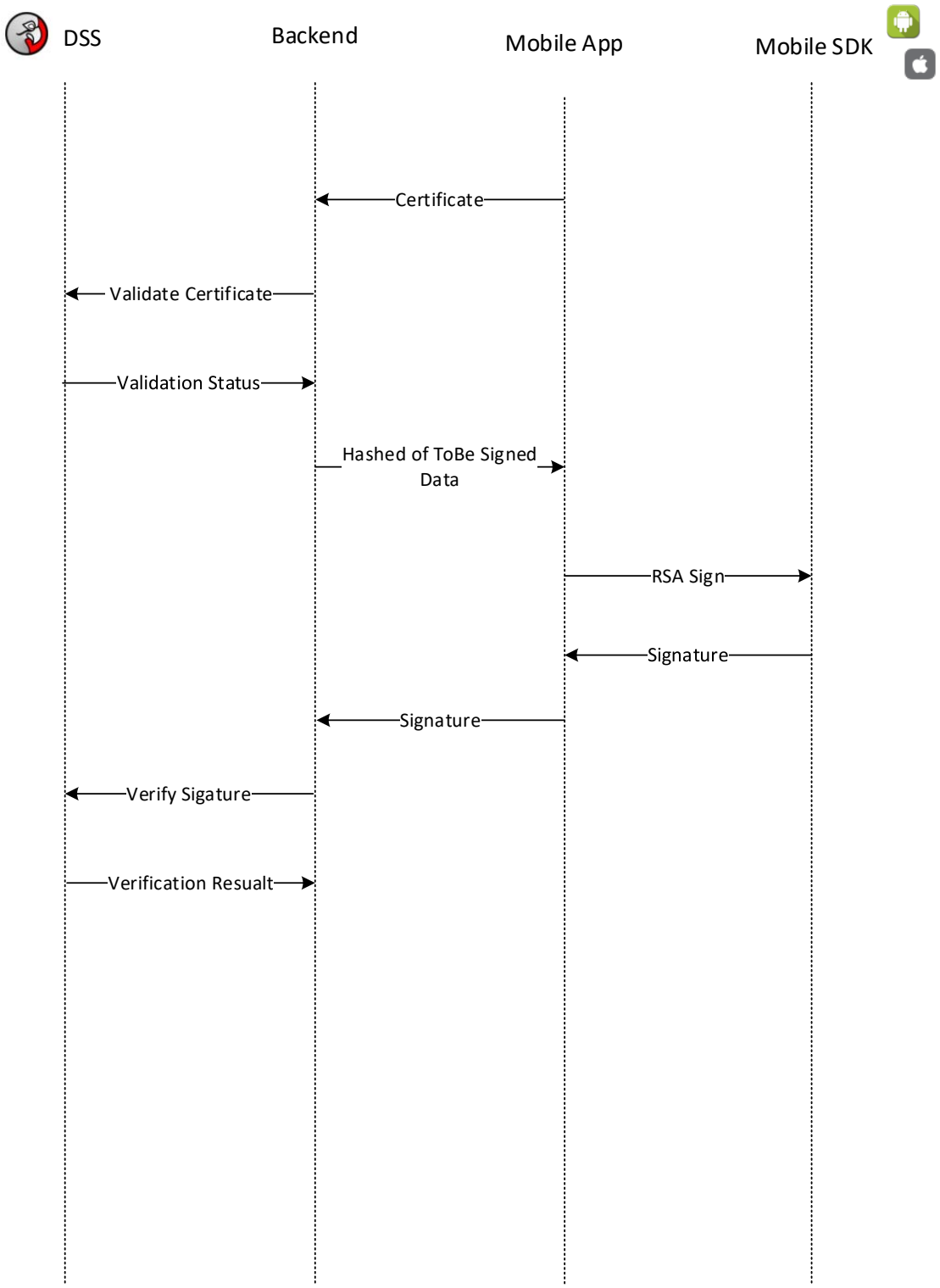


فهرست توابع موردنیاز جهت امضای RSA با دستینه:

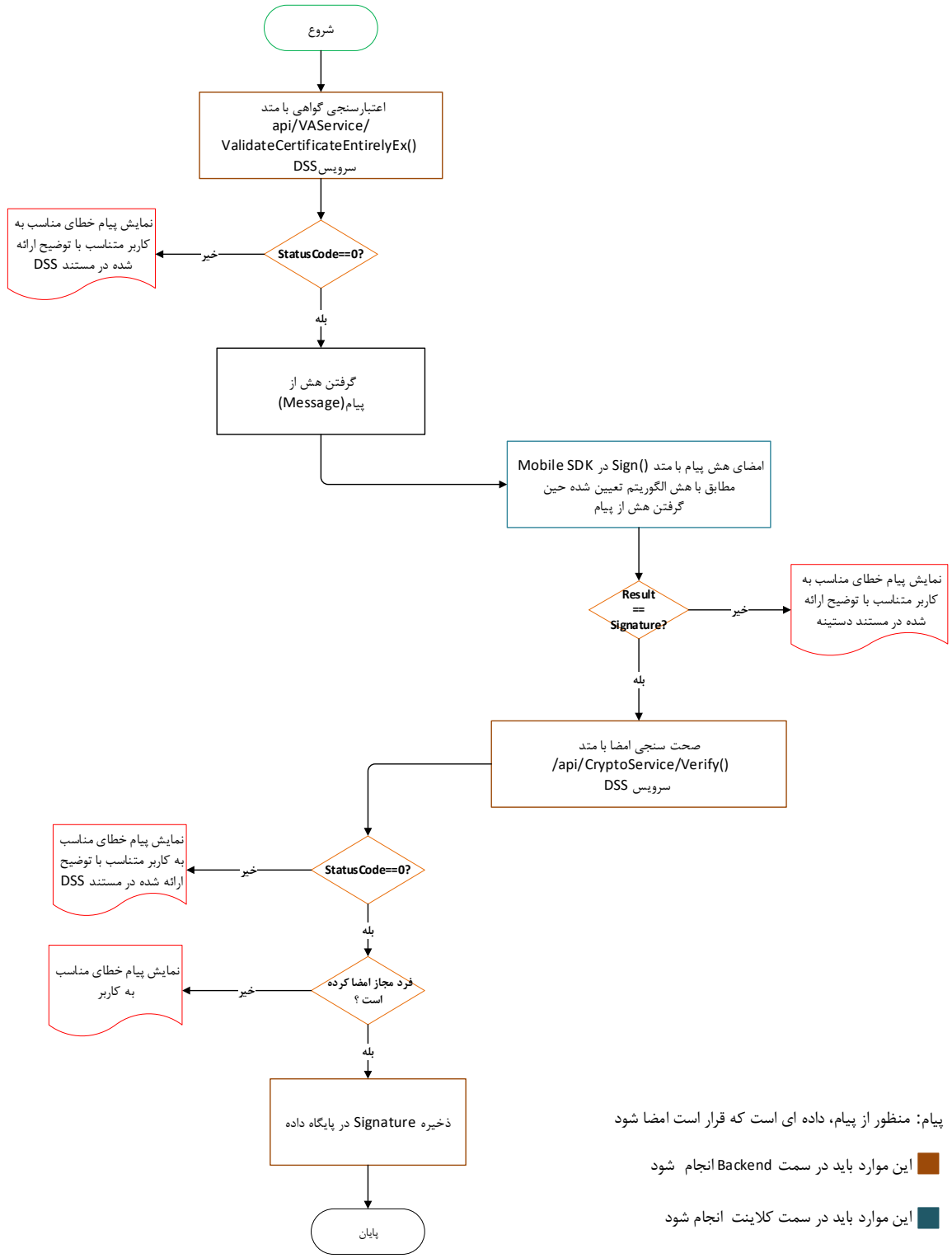
۱. انتخاب گواهی موردنظر جهت امضا با فراخوانی متد SelectCertificateFromTokenByUI() در دستینه
۲. دریافت گواهی انتخاب شده با فرمت Base64 توسط متد GetSelectedCertificate() در دستینه
۳. اعتبارسنجی گواهی از طریق متد /api/VAService/ValidateCertificateEntirelyEx در سرویس DSS
٤. ایجاد هش (Hash) از پیام
۵. استفاده از متد Sign() دستینه با پارامترهای ورودی زیر جهت امضای هش پیام :
   - string data : هش پیام با فرمت Base64
   - String hashAlg : مطابق با هش الگوریتم استفاده شده حین ایجاد هش از پیام، مقداردهی شود
۶. صحت سنجی امضا با فراخوانی متد /api/CryptoService/Verify در سرویس DSS

۵٫۱۸٫۲امضای **RSA** توسط **SDK** موبایل

نمودار توالی امضای RSA توسط SDK موبایل



فلوچارت توالی امضای RSA توسط SDK موبایل



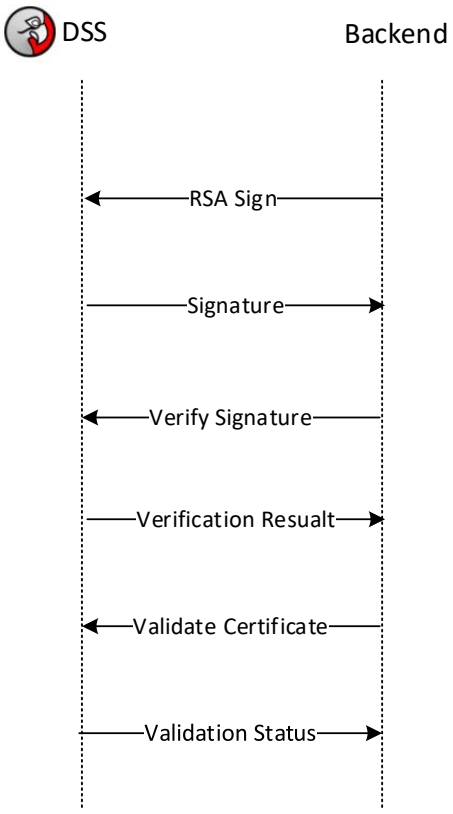
فهرست توابع موردنیاز جهت امضای RSA با SDK موبایل:

۱. اعتبارسنجی گواهی امضا از طریق متد /api/VAService/ValidateCertificateEntirelyEx در سرویس DSS
۲. ایجاد هش (Hash) از پیام
۳. استفاده از متد ()Sign در Mobile SDK با پارامترهای ورودی زیر جهت امضای هش پیام :

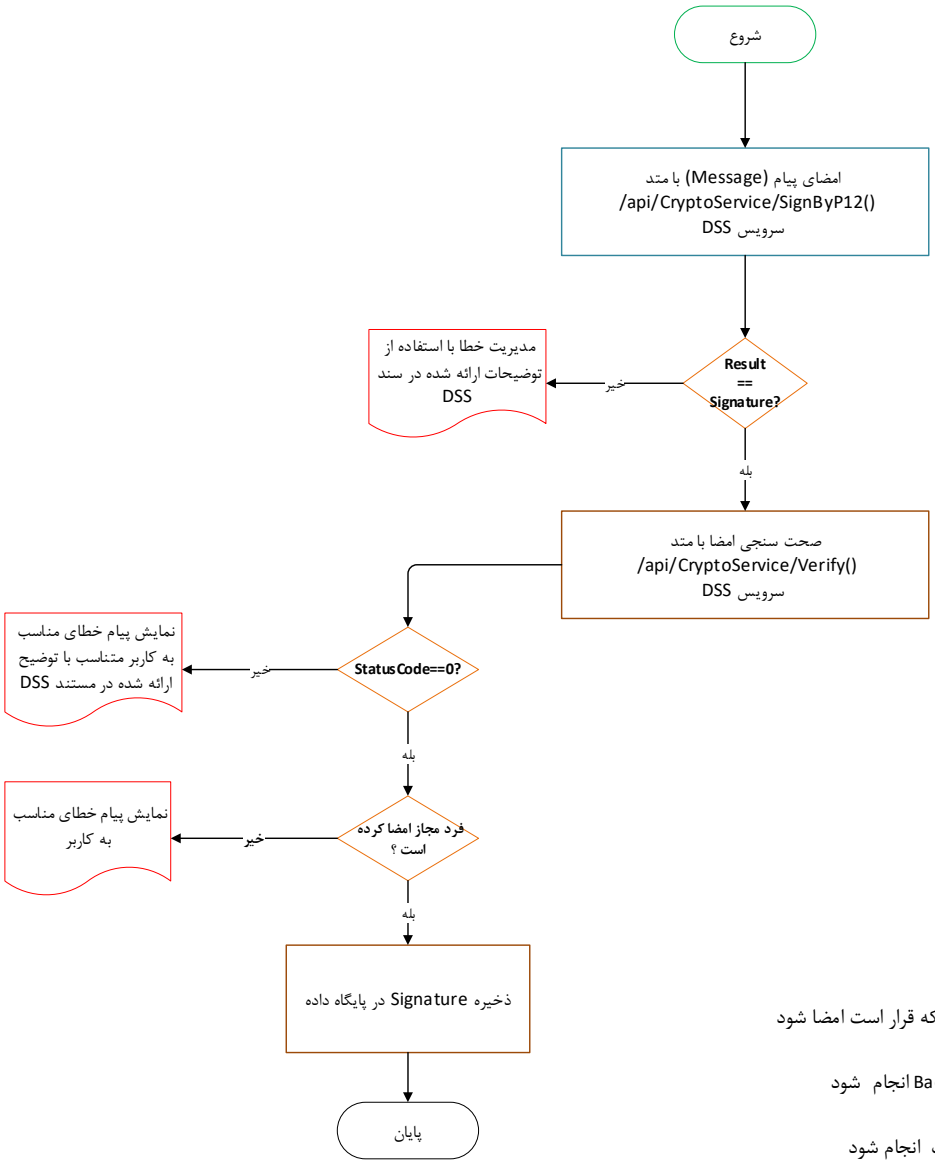
- string message : هش پیام با فرمت Base64
- string privateKeyName : نام کلید خصوصی (مطابق با توضیحات ارائه شده در سند SDK موبایل مقداردهی شود)
- string hashAlg : مطابق با هش الگوریتم استفاده شده حین ایجاد هش از پیام، مقداردهی شود
- string pin : پین کلید خصوصی (مطابق با توضیحات ارائه شده در سند SDK موبایل مقداردهی شود)
- صحت سنجی امضا با فراخوانی متد /api/CryptoService/Verify در سرویس DSS

۵/۱۸/۳ امضای **RSA** توسط توکن نرم افزاری

نمودار توالی امضای RSA توسط توکن نرم افزاری



فلوچارت توالی امضای RSA توسط توکن نرم افزاری



پیام: منظور از پیام، داده ای است که قرار است امضا شود

این موارد باید در سمت Backend انجام شود

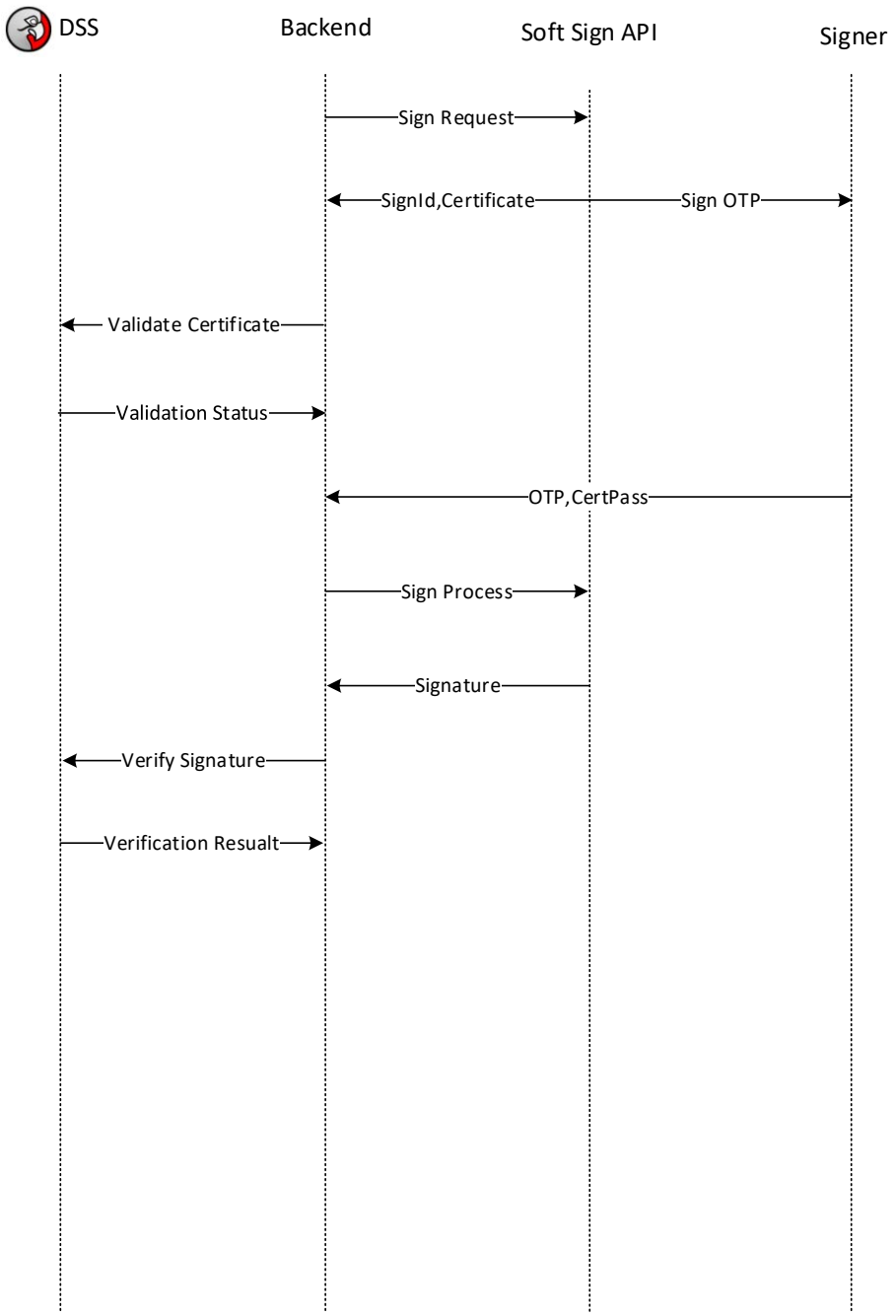
این موارد باید در سمت کلاینت انجام شود

فهرست توابع موردنیاز جهت امضای RSA توسط توکن نرم افزاری:

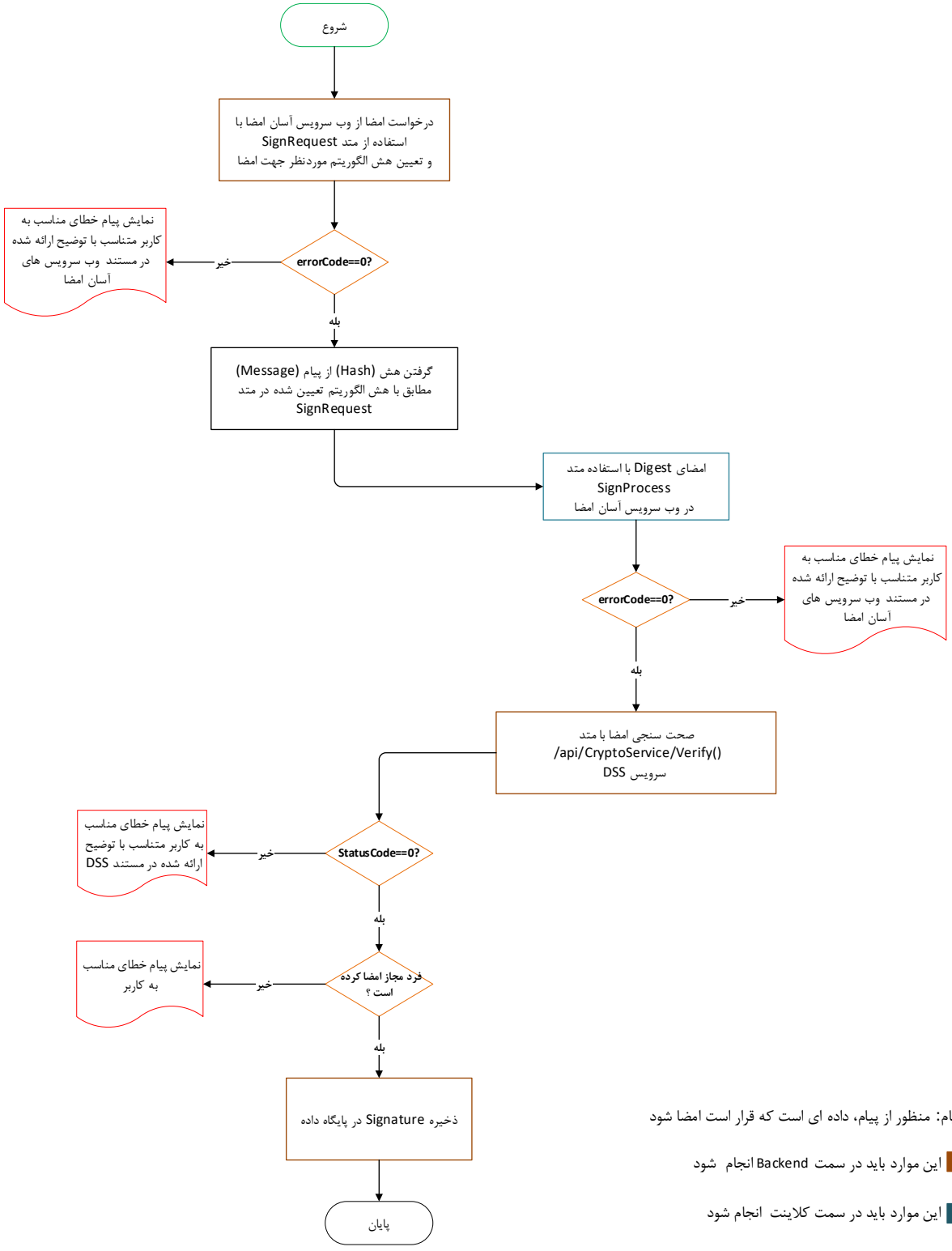
۱. امضای پیام با فراخوانی متد /api/CryptoService/SignByP12 در سرویس DSS
۲. صحت سنجی امضا با فراخوانی متد /api/CryptoService/Verify در سرویس DSS

۵٫۱۸٫۴امضای **RSA** توسط ابر آسان امضا

نمودار توالی امضای RSA توسط ابر آسان امضا



فلوچارت توالی امضای RSA توسط ابر آسان امضا



فهرست توابع موردنیاز جهت امضای RSA توسط ابر اسان امضا:

۱. درخواست امضا با استفاده از عمل SignRequest در وب سرویس های آسان امضا
۲. ایجاد هش (Hash) از پیام مطابق با هش الگوریتم تعیین شده حین استفاده از عمل SignRequest
۳. امضای هش پیام با استفاده از عمل SignProcess در وب سرویس های آسان امضا با پارامترهای ورودی زیر:
   - signId : شناسه امضا (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقداردهی شود)
   - dataforsign : هش پیام با فرمت base64
   - password : رمز گواهی (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقداردهی شود)
   - otp : رمز امضا (مطابق با توضیحات ارائه شده در سند وب سرویس های آسان امضا مقدادهی شود)
   - pkcs1support : False
۴. صحت سنجی امضا با فراخوانی متد /api/CryptoService/Verify در سرویس DSS

api/CryptoService/SignByP12	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CryptoSignByP12Request<string>
	{
	"data": "string",
	پیام موردنظر جهت امضا با فرمت Base64
	"certificate": "string",
	گواهی امضاکننده (فایل P12) با فرمت Base64
	"hashAlgorithm": hashAlgorithm, جدول شماره ۴
	الگوریتم درهم سازی موردنظر جهت استفاده در عملیات امضا
	"password": "string" توجه
	رمز گواهی امضاکننده با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	امضای پیام با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

درصورتی که فایل p12 یا pfx، فاقد رمز باشد باید مقدار certificatePassword برابر با null درنظرگرفته شود.

۵/۱۹ تصدیق امضای الکترونیک

api/CryptoService /Verify	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0
Schema	VerifyRequest<string> {

	"data": "string", محتوای اولیه (قبل از امضا) در قالب Base64 "signature": "string", محتوای امضا با فرمت Base64 "certificate": "string", گواهی امضاکننده با فرمت Base64 "hashAlgorithm": HashAlgorithm جدول شماره ۴ الگوریتم درهم سازی استفاده شده در عملیات امضا }
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی امضا

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

۵/۲۰ اعتبار سنجی سند XML امضا شده

/api/CryptoService/XMLVerify	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	dataUrl: string سند XML امضا شده با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتیجه صحت سنجی امضا

	{
	"error": "string",
	"result": bool,
	"statusCode": number => 200 Success / 500 Error
	}

۵/۲۱ رمزگذاری

api/CryptoService/Encrypt	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CryptoRequest<string>
	{
	"cipher ": "string",
	اطلاعات موردنظر برای رمزنگاری با فرمت Base64
	"certificate": "string",
	گواهی مورداستفاده برای عملیات رمزنگاری با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	اطلاعات رمزشده با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

api/CryptoService/CmsEncrypt	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CmsEncryptionRequest<string>
	{
	"message": "string",
	اطلاعات موردنظر برای رمزنگاری با فرمت Base64
	"certificates": IEnumerable<string>
	لیست گواهینامه های مورداستفاده برای عملیات رمزنگاری با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	اطلاعات رمزشده با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۲۲ رمزگشایی

api/CryptoService/Decrypt	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	CryptoRequest<string>
	{
	"cipher ": "string",
	اطلاعات رمزشده با فرمت Base64
	"certificate": "string",
	گواهی مورداستفاده برای عملیات رمزنگاری با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	اطلاعات رمزگشایی شده با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

api/CryptoService/CmsDecrypt	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Body	cipher: string اطلاعات رمزشده با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest

200 Schema	اطلاعات رمزگشایی شده با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۵/۲۳ رمزگذاری متقارن

api/CryptoService/SymmetricEncrypt	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	SymmetricEncryptRequest<string>
	{
	"key": "string",
	کلید رمزگذاری با فرمت Base64
	"iv": "string",
	بردار آغازین با فرمت Base64
	"symmetricAlgorithm": SymmetricAlgorithms, جدول شماره ۸
	الگوریتم مورد استفاده در عملیات رمزگذاری
	"data": "string"
	اطلاعات موردنظر برای عملیات رمزگذاری با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	اطلاعات رمزگذاری شده با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۵/۲۴ رمزگشایی متقارن

api/CryptoService/SymmetricDecrypt	
Request	
Method	Post

Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	SymmetricDecryptRequest<string>
	{
	"key": "string",
	کلید رمزگشایی با فرمت Base64
	"iv": "string",
	بردار آغازین با فرمت Base64
	"symmetricAlgorithm": SymmetricAlgorithms, جدول شماره ۸
	الگوریتم استفاده شده عملیات رمزگذاری
	"cipher": "string"
	اطلاعات رمزشده با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	اطلاعات رمزگشایی شده با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۵/۲۵استخراج پیام از قالب CMSAttached

api/CryptoService/CmsExtractAttachedMessage	
Request	
Method	Post
Header	Content-Type: multipart/form-data Accept: text/plain api-version : 2.0 or 3.0
Schema	enveloped: string محتوای امضاشده (CMS) با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	پیام استخراج شده از قالب CMS با فرمت Base64
	{ "error": "string", "result": "string", "statusCode": number => 200 Success / 500 Error

	}

۵٫۲۶استخراج Thumbprint از فایل گواهی

api/CryptoService/CertificateThumbprint	
Request	
Method	Post
Header	Content-Type: application/octet-stream Accept: text/plain api-version : 2.0 or 3.0
Schema	base64Certificate: string گواهی موردنظر با فرمت Base64
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	Thumbprint گواهی با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۵٫۲۷ادغام امضاهای CMS

۵٫۲۷٫۱ادغام امضاهای (**Detached**)**CMS**

به منظور ادغام امضاهای CMS(Detached)، ۲ متد در کلاس Crypto درنظرگرفته شده است.

شایان ذکر است که در هر ۲ متد درصورتی که بیش از ۲ امضاکننده برای امضای یک محتوا وجود دارد ابتدا برای ادغام امضای نفر اول و دوم اقدام نمایید و سپس خروجی تابع را به عنوان پارمتر اول با امضای نفر بعدی به عنوان پارمتر دوم، ارسال کنید و همین روال را تا اتمام تمامی امضاها تکرار نمایید.

توجه به این نکته الزامیست که پیام امضاشده در تمامی امضاهای صورت گرفته باید یکسان باشد. در غیر اینصورت صحت‌سنجی امضای ادغام شده با خطا مواجه می‌گردد.

api/CryptoService/MergeCmsSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	MergeCMSSignRequest<string>
	{
	"Message": "string", توجه
	پیام امضاشده، با فرمت string
	"Cms1": "string",
	امضای اول در قالب CMS(Detached)(شامل امضا و گواهی امضا)، با فرمت string
	"Cms2": "string"
	امضای دوم در قالب CMS(Detached)(شامل امضا و گواهی امضا)، با فرمت string
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	امضاهای ادغام شده در قالب CMS(Detached)(شامل امضا و گواهی امضا)، با فرمت byte[]

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه

ارسال پارامتر Message اجباری نمی‌باشد ولی :

۱. درصورت ارسال پارامتر Message ابتدا صحت‌سنجی امضاها با استفاده از متد CmsVerify انجام شده و در صورت معتبر بودن امضاها، عملیات ادغام انجام می‌شود.

۲. درصورت عدم ارسال پارامتر Message، امکان صحت سنجی فراهم نمی‌باشد لذا امضاها بدون ارزیابی، ادغام می‌شوند.

۵٫۲۷٫۲ ادغام امضاهای CMS(Attached)

به منظور ادغام امضاهای CMS(Attached)، از ۲ متد در کلاس Crypto استفاده می‌گردد.

شایان ذکر است که در هر ۲ متد درصورتی که بیش از ۲ امضاکننده برای امضای یک محتوا وجود دارد ابتدا برای ادغام امضای نفر اول و دوم اقدام نمایید و سپس خروجی تابع را به عنوان پارمتر اول با امضای نفر بعدی به عنوان پارمتر دوم، ارسال کنید و همین روال را تا اتمام تمامی امضاها تکرار نمایید.

api/CryptoService/MergeCmsAttachSign	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain

	api-version : 2.0 or 3.0

Schema	CMSAttachSignRequest<string>
	{
	"currentMessgaeSignatureCertificate": "string",
	امضای جاری در قالب CMS(Attached)(شامل محتوا، امضا و گواهی امضا)، با فرمت Base64
	"nextMessgaeSignatureCertificate": "string"
	امضای بعدی در قالب CMS(Attached)(شامل محتوا، امضا و گواهی امضا)، با فرمت Base64
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	امضاهای ادغام شده در قالب CMS(Attached)(شامل محتوا، امضا و گواهی امضا)، با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۶ سرویس ورود به سیستم(LoginService)

سرویس Login به منظور دسته‌بندی توابع مورد نیاز جهت اجرای روال شناسایی کاربر و ورود به سیستم، ایجاد شده است. متدهای موجود در این سرویس ، توابع مورد نیاز جهت تولید Challenge و تصدیق هویت کاربر می‌باشد.

در فرآیند تشخیص هویت کاربر، سرور با استفاده از متدLoginChallenge از کلاس Login یک challenge تولید نموده و به سمت کاربر ارسال می‌نماید. در سمت کاربر challenge مربوطه با استفاده ازکلید خصوصی موجود در توکن کاربر امضا شده و رشته نهایی که امضای دیجیتال کاربر نیز در آن وجود دارد برای سرور ارسال خواهد شد. سپس سرور بر اساس کلید عمومی موجود در امضای الصاقی به رشته دریافت شده از سمت کاربر، هویت وی را تشخیص داده و در صورت تایید با توجه به سطوح دسترسی تعریف شده در سیستم، ورود کاربر را به سیستم میسر ساخته و منابع مورد درخواست را در اختیار وی قرار خواهد داد.

۶٫۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

<Http protocol>://DSSUrl/API/LoginService

۶٫۲ تولید Challenge

api/LoginService/LoginChallenge	
Request	
Method	Post
Header	Accept: text/plain api-version : 2.0 or 3.0
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	challenge با طول ۲۰ با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۶٫۳ تصدیق هویت کاربر

api/LoginService/Authenticate	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	AuthRequest <string>
	{
	"random": string,
	challenge با فرمت Base64
	"cmsSignature": string, توجه ۱
	challenge امضاشده توسط کاربر با فرمت Base64
	"loginProfile": string توجه ۲
	نام پروفایل مورد استفاده جهت دسترسی سرویس Login
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نتایج بازگشتی از بررسی تصدیق هویت کاربر
	{ "error": string, "result": AuthResponse<string> {

	"authenticationResult": AuthenticationResult, جدول شماره ۶ وضعیت تصدیق هویت کاربر "certificate": string گواهی امضا کاربر با فرمت Base64 }, "statusCode": number => 200 Success / 500 Error }

توجه

۱. challenge ابتدا باید از طریق گواهی کاربر مطابق با استاندارد PKCS#7 امضا شود و سپس به عنوان مقدار پارامتر cmsSignature در متد Authenticate قرارگیرد.

۲. جهت اطلاع از نحوه مقداردهی به پارامتر loginProfile با تیم استقرار سرویس هماهنگ شوید.

۷ سرویس ارتباط با مخزن یا دایرکتوری کلید عمومی(PKDSrvice)

در PKDService متدهایی به منظور دسته‌بندی توابع موردنیاز جهت برقراری ارتباط با دایرکتوری کلید عمومی قرار داده شده است. متدهای موجود در این سرویس، توابع مورد نیاز جهت دریافت گواهی خاص از دایرکتوری کلید عمومی و یا قراردهی گواهینامه بر روی این دایرکتوری را ارائه می‌دهند.

۷/۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

<Http protocol>://DSSUrl/API/ PKDService

۷/۲ دریافت گواهینامه از مخزن

api/PKDService/DownloadCertificate	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	PKDClientRequest
	{
	"path": "string", توجه ۱
	آدرس نوده مربوطه بعد از BaseDN در قالب استاندارد X500
	"pkdProfile": "string" توجه ۲
	نام پروفایل جهت ارتباط با دایرکتوری کلید عمومی (PKD)
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از گواهی‌های دریافت‌شده از دایرکتوری کلید عمومی

	{
	"error": "string",
	"result": IEnumerable<byte[]>,
	"statusCode": number => 200 Success / 500 Error
	}

api/PKDService/DownloadCertificateAsBase64	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	PKDClientRequest
	{
	"path": "string", توجه ۱
	آدرس نوده مربوطه بعد از BaseDN در قالب استاندارد X500
	"pkdProfile": "string" توجه ۲
	نام پروفایل جهت ارتباط با دایرکتوری کلید عمومی (PKD)
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فهرستی از گواهی‌های دریافت‌شده از دایرکتوری کلید عمومی

	{
	"error": "string",
	"result": IEnumerable<string>
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. برای مثال با فرض براینکه مقدار BaseDN در pkdProfile برابر با dc=End Entity,dc=g3,c=ir قرارگیرد با وارد کردن آدرس زیر در پارامتر path، گواهی مربوطه دانلود خواهد شد.

"cn=PKI Private Intermediate Bronze CA - G3"

۲. جهت اطلاع از نحوه مقداردهی به پارامتر pkdProfile با تیم استقرار سرویس هماهنگ شوید.

۷/۳ دریافت CRL از مخزن

/api/PKDService/DownloadCRL	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	PKDClientRequest
	{
	"path": "string", توجه ۱
	آدرس نوده مربوطه بعد از BaseDN در قالب استاندارد X500
	"pkdProfile": "string" توجه ۲
	نام پروفایل جهت ارتباط با دایرکتوری کلید عمومی (PKD)
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فایل CRL دریافت‌شده از دایرکتوری کلید عمومی با فرمت byte[]

	{
	"error": "string",
	"result": byte[],
	"statusCode": number => 200 Success / 500 Error
	}

api/PKDService/DownloadCRLAsBase64	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	PKDClientRequest
	{
	"path": "string", توجه ۱
	آدرس نوده مربوطه بعد از BaseDN در قالب استاندارد X500
	"pkdProfile": "string" توجه ۲
	نام پروفایل جهت ارتباط با دایرکتوری کلید عمومی (PKD)
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	فایل CRL دریافت‌شده از دایرکتوری کلید عمومی با فرمت Base64

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. برای مثال با فرض براینکه مقدار BaseDN در pkdProfile برابر با dc=End Entity,dc=g3,c=ir قرارگیرد با وارد کردن آدرس زیر در پارامتر path، فایل CRL مربوطه دانلود خواهد شد

"cn=PKI Private Intermediate Bronze CA - G3"

۲. جهت اطلاع از نحوه مقداردهی به پارامتر pkdProfile با تیم استقرار سرویس همانگ شوید.

۷/۴ دریافت لیست دایرکتوری

api/PKDService/SubDirectoryList	
Request	
Method	Post
Header	Content-Type: application/json Accept: text/plain api-version : 2.0 or 3.0

Schema	PKDClientRequest
	{
	"path": "string", توجه ۱
	آدرس نود مربوطه بعد از BaseDN در قالب استاندارد X500
	"pkdProfile": "string" توجه ۲
	نام پروفایل جهت ارتباط با دایرکتوری کلید عمومی (PKD)
	}

Response	
Status Code	200 Success / 400 BadRequest
200 Schema	لیست کلیه نودهای زیرمجموعه

	{
	"error": "string",
	"result": "string[]",
	"statusCode": number => 200 Success / 500 Error
	}

توجه:

۱. برای مثال با فرض براینکه مقدار BaseDN در pkdProfile برابر با dc=End Entity,dc=g3,c=ir قرارگیرد با وارد کردن آدرس زیر در پارامتر path، لیست کلیه نودهای زیرمجموعه dc=Level 1 نمایش داده می شود

"dc=Level 1"

۲. جهت اطلاع از نحوه مقداردهی به پارامتر pkdProfile با تیم استقرار سرویس هماهنگ شوید.

۸ سرویس دریافت نسخه(ApiVersion)

۸/۱ آدرس فراخوانی سرویس

امکان فراخوانی متدها به صورت REST FUL API در آدرس زیر مهیا شده است:

https://IP/API/ApiVersion

۸/۲ دریافت نسخه

api/ApiVersion/GetVersion	
Request	
Method	Get
Header	Accept: text/plain api-version : 2.0 or 3.0
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نسخه‌ی سرویس به همراه آدرس ip فراخواننده سرویس

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

api/ApiVersion/Version	
Request	
Method	Post
Header	Accept: text/plain api-version : 2.0 or 3.0
Response	
Status Code	200 Success / 400 BadRequest
200 Schema	نسخه‌ی سرویس به همراه آدرس ip فراخواننده سرویس

	{
	"error": "string",
	"result": "string",
	"statusCode": number => 200 Success / 500 Error
	}

۹ پیوست

CertificateStatus	
کد	شرح
0	Good
1	Revoked
2	Unknown

جدول شماره ۱

RevocationReason	
کد	شرح
1	keyCompromise
2	cACompromise
3	affiliationChanged
4	superseded
5	cessationOfOperation
6	certificateHold
7	privilegeWithdrawn
8	aACompromise

جدول شماره ۲

CertificateValidationStatus	
کد	شرح
0	CertificateValidationOK
1	PeriodValidationFailed
2	ChainValidationFailed
3	IntgrityValidationFailed
4	KeyUsageValidationFailed
5	OCSPValidationRevoked
6	OCSPValidationUnKnown
7	CRLValidationRevoked
8	CRLAndOCSPValidationError
9	OCSPValidationException
10	CRLValidationUnKnown
11	CRLAndOCSPValidationUnknown

جدول شماره ۳

HashAlgorithm	
کد	شرح
0	SHA1
1	SHA256
2	SHA384
3	SHA512

جدول شماره ۴

CMSVerifyValidationStatus	
کد	شرح
0	VerificationOK
1	CertPeriodValidationFailed
2	CertChainValidationFailed
3	CertIntgrityValidationFailed
4	CertKeyUsageValidationFailed
5	CertOCSPValidationRevoked
6	CertOCSPValidationUnKnown
7	CertCRLValidationRevoked
8	CertCRLAndOCSPValidationFailed
9	VerificationFailed
10	CMSDataNotAttached
11	CMSFromatIncorrect
12	CertPeriodAndTimeMismatch
13	SignitureNotFound
14	InvalidSignDateTime

جدول شماره ۵

AuthenticationResult	
کد	شرح
0	UnKnown
1	Athenticated
2	SignatureVerificationError
3	CertificateValidationError
4	CertificateCrlCheckError
5	CertificateOcspCheckError
6	CertificateKeyUsageError

7	CertificateEnhancedKeyUsageError

جدول شماره ۶

KeyUsage	
کد	شرح
0	EncipherOnly
1	CrlSign
2	KeyCertSign
3	KeyAgreement
4	DataEncipherment
5	KeyEncipherment
6	NonRepudiation
7	DigitalSignature
8	DecipherOnly

جدول شماره ۷

SymmetricAlgorithms	
کد	شرح
0	AES_CBC128
1	AES_CBC256
2	AES_CBC192
3	AES_ECB128
4	AES_ECB256
5	AES_ECB192

جدول شماره ۸

CertificationLevel	
کد	شرح
0	NOT_CERTIFIED
1	CERTIFIED_NO_CHANGES_ALLOWED
2	CERTIFIED_FORM_FILLING
3	CERTIFIED_FORM_FILLING_AND_ANNOTATIONS

جدول شماره ۹