公的個人認証サービス

利用者クライアントソフト API 仕様書

【カード AP ライブラリ PKCS#11 編】

第１.１版

公的個人認証サービス　指定認証機関

財団法人　自治体衛星通信機構

変更履歴

版数	変更内容
1.0 版	新規作成
1.1 版	Windows XP SP2 対応に伴い表 1(2 頁)のプラットホームを追加

－ 目次 －

第１章 はじめに １
第２章 ドキュメント体系 １
第３章 動作環境 ２
第４章 機能仕様 ３
第１節 ソフトウェア構成図 3
第２節 実現可能な機能の一覧 4
第５章 API仕様 ５
第１節 サポートAPI一覧 5
第２節 サポートAPI仕様詳細 6
第３節 構造体仕様 ２０
第４節 コーリングシーケンス ２２
第６章 その他 ３０
第１節 ライブラリのロード方法 ３０

第１章　はじめに

利用者クライアントソフトにおけるカード AP ライブラリは、以下の機能を実現するための Application Program Interface(以下、API)を提供する。

- 証明書取得機能
- 電子署名生成機能
- 電子署名検証機能

以降、本書ではカード AP ライブラリのうち、PKCS#11 の API 仕様について説明する。

第２章　ドキュメント体系

利用者クライアントソフトのドキュメント体系図を以下に示す。本書は以下の体系図の網掛け部分に該当する。

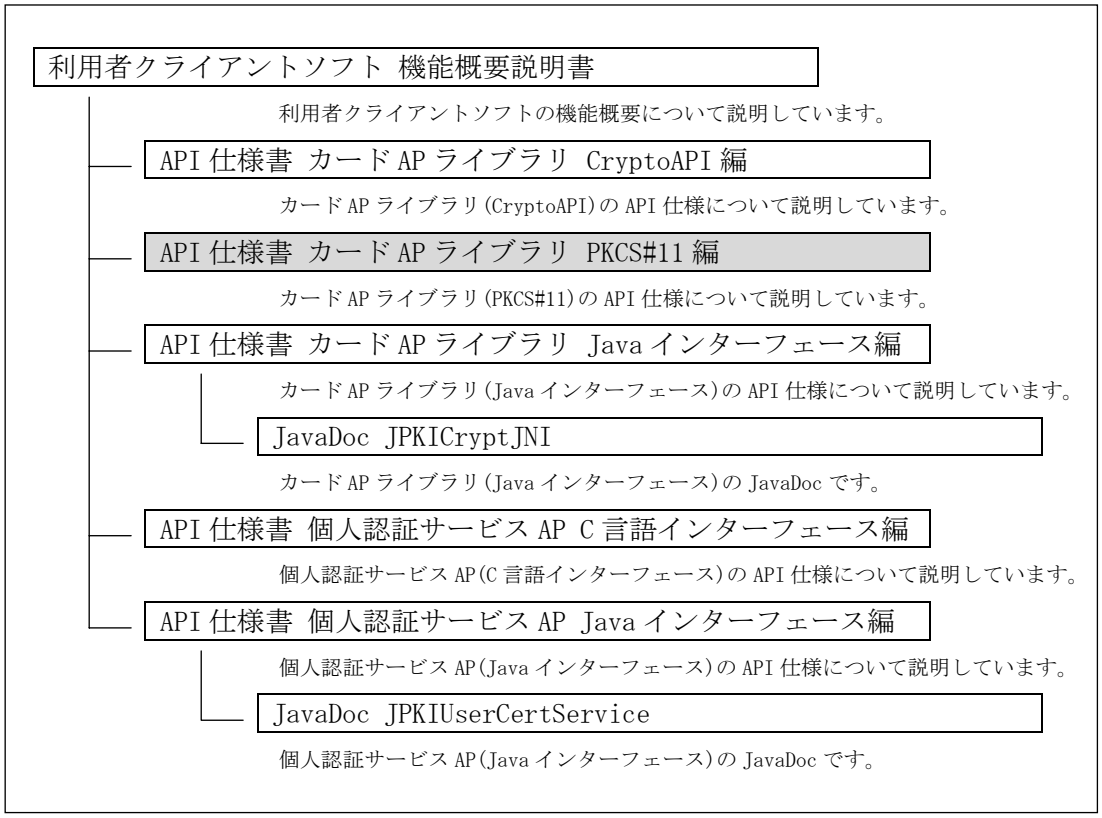


図 2.1 ドキュメント体系図

第３章　動作環境

カード AP ライブラリの動作環境は以下の通りとする。

表 1 動作環境

項目	条件
プラットフォーム	Windows98 Second Edition(※1) Windows Millennium Edition(※1) WindowsNT4.0 ServicePack6a(※1) Windows2000 ServicePack2 Windows2000 ServicePack3 Windows2000 ServicePack4 WindowsXP ServicePack1 WindowsXP ServicePack2
Web ブラウザ(※2)	Internet Explorer5.5 ServicePack2 Internet Explorer6 ServicePack1
IC カード	公的個人認証サービスカードアプリケーションを搭載し、公的個人認証サービスの電子証明書が格納されたICカードとする。
IC カードリーダライタ	以下の条件を満たす IC カードリーダライタとする。(「適合性検証済み IC カードリーダライター一覧」を参照のこと。) ・ IC カードのインターフェース(非接触型、接触非接触両対応型)に対応していること ・ USB や RS-232C など、パソコンに接続するためのインターフェースを有すること ・ IC カードリーダライタと通信するためのドライバソフトウェアが提供されていること ・ IC カードの搬送方式が手動挿入/手動排出タイプまたは自動挿入/自動排出タイプであること ・ IC カードを挿入するスロットの数は 1 つとし、1 度に挿入できる IC カードは 1 枚であること

※1 IC カードの利用のため、Microsoft Smart Card Base Components が必要になる。

※2 暗号機能等の利用のため、Internet Explorer5.5 ServicePack2 もしくは Internet Explorer6 ServicePack1 が必要になる。

第４章　機能仕様

第１節　ソフトウェア構成図

本仕様書では、利用者クライアントソフトのうち、下図の太枠に示すカード AP ライブラリ(PKCS#11)の仕様をまとめる。

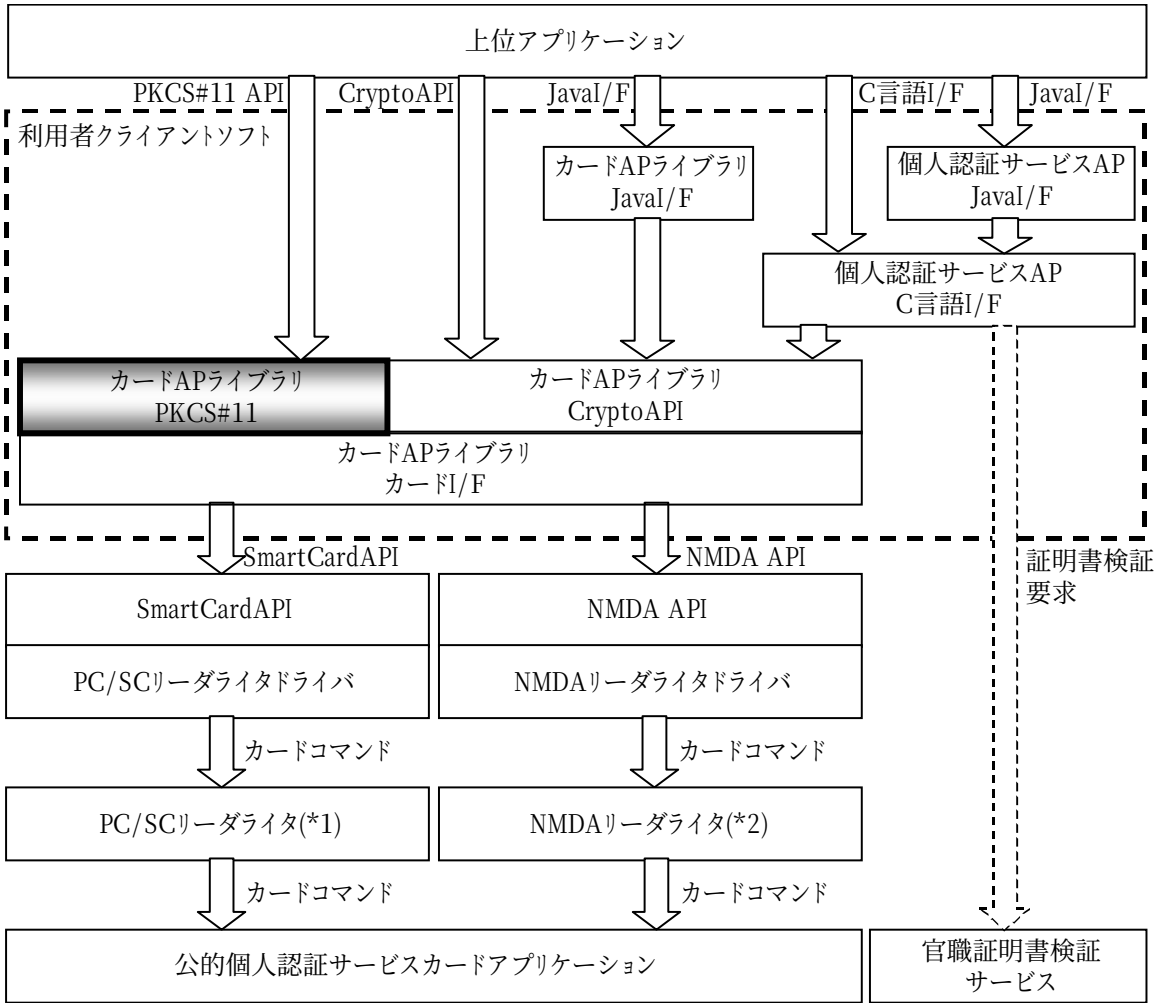


図 2 ソフトウェア構成図

*1 Personal Computer/Smart Cardの略。Microsoft社等のワーキンググループが推進する、Windows環境におけるICカード利用のための統一規格(PC/SC規格)に対応したICカードリーダライタのことを指す。

*2 New Media Development Associationの略。(財)ニューメディア開発協会「IT装備都市研究事業 リーダライタ共通インターフェース仕様書 1.1 版[平成 14 年 5 月 29 日]」に対応したICカードリーダライタのことを指す。

第２節　実現可能な機能の一覧

カード AP ライブラリ(PKCS#11)で実現可能な機能の一覧を表 2に示す。

表 2 実現可能な機能の一覧

NO	機能	概要
1	証明書取得	IC カードに格納された電子証明書(利用者証明書、都道府県知事の自己署名証明書)を取得する。
2	署名生成	署名対象データからハッシュ値を計算し、IC カードに格納された利用者秘密鍵を使用して電子署名を生成する。
3	署名検証	検証対象データからハッシュ値を計算し、ハッシュ値、電子署名、公開鍵を使用して電子署名を検証する。
4	繰り返し署名生成	NO2 の処理を繰り返し実行し、複数の署名対象データに対する電子署名を生成する。
5	繰り返し署名検証	NO3 の処理を繰り返し実行し、複数の電子署名を検証する。

第５章　API 仕様

第１節　サポート API 一覧

カード AP ライブラリ(PKCS#11)のサポート API の一覧を表 3に示す。

表 3 サポート API 一覧

NO	API 名	概要
1	C_GetFunctionList	関数ポインタリストを取得する。
2	C_Initialize	PKCS#11 ライブラリを初期化する。
3	C_Finalize	PKCS#11 ライブラリを終了する。
4	C_GetInfo	ライブラリ情報を取得する。
5	C_GetSlotList	スロットリストを取得する。
6	C_GetSlotInfo	スロット情報を取得する。
7	C_GetTokenInfo	トークン情報を取得する。
8	C_GetMechanismList	サポートメカニズム（アルゴリズム）を取得する。
9	C_GetMechanismInfo	メカニズム（アルゴリズム）情報を返す。
10	C_OpenSession	セッションを確立する。
11	C_CloseSession	セッションを切断する。
12	C_CloseAllSessions	すべてのセッションを切断する。
13	C_GetSessionInfo	セッション状態を取得する。
14	C_Login	トークンをログイン状態にする。
15	C_Logout	トークンをログアウト状態にする。
16	C_FindObjectsInit	オブジェクトの検索を開始する。
17	C_FindObjects	オブジェクトの検索を行う。
18	C_FindObjectsFinal	オブジェクトの検索を終了する。
19	C_GetAttributeValue	オブジェクトの属性値を取得する。
20	C_SignInit	署名処理を初期化する。
21	C_Sign	データに署名を行う。
22	C_DigestInit	ダイジェスト作成を開始する。
23	C_DigestUpdate	ダイジェストを作成する。
24	C_DigestFinal	ダイジェスト作成を終了する。
25	C_VerifyInit	署名検証を開始する。
26	C_Verify	署名値を検証する。
27	C_CreateObject	公開鍵オブジェクトを作成する。
28	C_DestroyObject	公開鍵オブジェクトを破棄する。

第２節　サポート API 仕様詳細

（1） C_GetFunctionList

API 名	C_GetFunctionList		
概要	関数ポインタリストを取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetFunctionList)(CK_FUNCTION_LIST_PTR_PTR ppFunctionList);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_FUNCTION_LIST_PTR_PTR	OUT	関数アドレスリストポインタ

（2） C_Initialize

API 名	C_Initialize		
概要	PKCS#11 ライブラリを初期化する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Initialize)(CK_VOID_PTR pReserved);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_VOID_PTR	IN	NULL ポインタを指定

（3） C_Finalize

API 名	C_Finalize		
概要	PKCS#11 ライブラリを終了する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Finalize)(CK_VOID_PTR pReserved);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_VOID_PTR	IN	NULL ポインタを指定

（4） C_GetInfo

API 名	C_GetInfo
概要	ライブラリ情報を取得する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetInfo)(CK_INFO_PTR pInfo);
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_INFO_PTR	IN/OUT	ライブラリ情報ポインタ
備考	取得可能なライブラリ情報は以下の通り。 CK_INFO::cryptokiVersion: PKCS11 規格バージョン：2.0 CK_INFO::manufacturerID: ライブラリ製造者名：「JPKI」 CK_INFO::description: ライブラリ記述文：「JPKI PKCS#11」 CK_INFO::libraryVersion: ライブラリバージョン:1.0		

（5） C_GetSlotList

API 名	C_GetSlotList		
概要	スロットリストを取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSlotList)(CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount);		
戻り値	CK_RV (CKR_OK: 成功 CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_BBOOL	IN	TRUE: カード有りのスロットリストを返す FALSE: 接続されているすべてのスロットリストを返す
	CK_SLOT_ID_PTR	IN/OUT	スロット ID リストポインタ
	CK_ULONG_PTR	IN/OUT	スロット ID リスト件数

（6） C_GetSlotInfo

API 名	C_GetSlotInfo		
概要	スロット情報を取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSlotInfo)(CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo);		
戻り値	CK_RV (CKR_OK: 成功 CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_SLOT_INFO_PTR	IN/OUT	スロット情報ポインタ

備考	取得可能なスロット情報は以下の通り。 CK_SLOT_INFO::hardwareVersion: スロットハードウェアバージョン: 0.0 CK_SLOT_INFO::firmwareVersion: スロットファームウェアバージョン: 0.0 CK_SLOT_INFO::slotDescription: スロット記述文: PCSC リーダ名称 CK_SLOT_INFO::manufacturerID: スロット製造者名: なし (0 バイトの文字列) CK_SLOT_INFO::flags: カード有無

(7) C_GetTokenInfo

API 名	C_GetTokenInfo		
概要	トークン情報を取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetTokenInfo)(CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo);		
戻り値	CK_RV (CKR_OK: 成功 CKR_TOKEN_NOT_PRESENT: カードが挿入されていないまたはカードが抜かれた CKR_TOKEN_NOT_RECOGNIZED: 不正な IC カードを検出した CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_TOKEN_INFO_PTR	OUT	トークン情報ポインタ

(8) C_GetMechanismList

API 名	C_GetMechanismList
概要	サポートメカニズム (アルゴリズム) を取得する。

関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismList)(CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_MECHANISM_TYPE_PTR	OUT	メカニズムタイプポインタ
	CK_ULONG_PTR	IN/OUT	メカニズムタイプ件数
備考	取得可能なサポートメカニズムは以下の通り。 Sign、Verify 用: CKM_RSA_PKCS Digest 用: CKM_SHA_1		

(9) C_GetMechanismInfo

API 名	C_GetMechanismInfo		
概要	メカニズム (アルゴリズム) 情報を返す。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismInfo)(CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_MECHANISM_TYPE	IN	メカニズムタイプ
	CK_MECHANISM_INFO_PTR	IN/OUT	メカニズム情報ポインタ
備考	設定する情報は以下の通り。 type = CKM_RSA_PKCS の場合 CK_MECHANISM_INFO::ulMinKeySize: 1024 CK_MECHANISM_INFO::ulMaxKeySize: 1024 CK_MECHANISM_INFO::flags: CKF_VERIFY | CKF_SIGN | CKF_HW type = CKM_SHA_1 の場合 CK_MECHANISM_INFO::ulMinKeySize: 0 CK_MECHANISM_INFO::ulMaxKeySize: 0 CK_MECHANISM_INFO::flags: CKF_DIGEST		

（１０） C_OpenSession

API 名	C_OpenSession		
概要	セッションを確立する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_OpenSession)(CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY Notify, CK_SESSION_HANDLE_PTR phSession);		
戻り値	CK_RV (CKR_OK: 成功 CKR_TOKEN_NOT_PRESENT: カードが挿入されていないまたはカードが抜かれた CKR_TOKEN_NOT_RECOGNIZED: 不正な IC カードを検出した CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_FLAGS	IN	0 または CKF_SERIAL_SESSION を指定
	CK_VOID_PTR	IN	NULL ポインタを指定
	CK_NOTIFY	IN	NULL ポインタを指定
	CK_SESSION_HANDLE_PTR	IN/OUT	セッションハンドルポインタ

（１１） C_CloseSession

API 名	C_CloseSession		
概要	セッションを切断する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_CloseSession)(CK_SESSION_HANDLE hSession);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

（１２） C_CloseAllSessions

API 名	C_CloseAllSessions

概要	すべてのセッションを切断する。		
関数インタ ーフェース	CK_DEFINE_FUNCTION(CK_RV, C_CloseAllSessions)(　CK_SLOT_ID slotID);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID

（１３） C_GetSessionInfo

API 名	C_GetSessionInfo		
概要	セッション状態を取得する。		
関数インタ ーフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSessionInfo)(　CK_SESSION_HANDLE　hSession, 　CK_SESSION_INFO_PTR pInfo);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションタイプ
	CK_SESSION_INFO_PTR	OUT	セッション状態ポインタ
備考	以下の状態を返す。 　ログインしていないとき: CKS_RO_PUBLIC_SESSION 　ログインしているとき: CKS_RO_USER_FUNCTIONS		

（１４） C_Login

API 名	C_Login
概要	トークンをログイン状態にする（証明書 DF を活性化する)。
関数インタ ーフェース	CK_DEFINE_FUNCTION(CK_RV, C_Login)(　CK_SESSION_HANDLE hSession, 　CK_USER_TYPE　　userType, 　CK_CHAR_PTR　　pPin, 　CK_ULONG　　ulPinLen);

戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_PIN_INCORRECT: パスワード指定誤り CKR_PIN_LOCKED: パスワードがロックされている CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_USER_TYPE	IN	ユーザタイプ
	CK_CHAR_PTR	IN	パスワード文字列ポインタ
	CK_ULONG	IN	パスワード文字列長

（１５） C_Logout

API 名	C_Logout		
概要	トークンをログアウト状態にする。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Logout)(CK_SESSION_HANDLE hSession);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

（１６） C_FindObjectsInit

API 名	C_FindObjectsInit
概要	オブジェクトの検索を開始する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsInit)(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_ATTRIBUTE_PTR	IN	属性テーブルポインタ
	CK_ULONG	IN	属性テーブル数
備考	以下の属性による検索を行う。 CKA_CLASS　CKO_CERTIFICATE または CKO_PRIVATE_KEY CKA_ID　Login で読み出したときに生成した番号 CKA_TOKEN　真 CKA_LABEL　証明書の名前 CKA_VALUE　証明書の値 CKA_MODULUS 利用者公開鍵の N CKA_PUBLIC_EXPONENT 利用者公開鍵の E		

（１７） C_FindObjects

API 名	C_FindObjects
概要	オブジェクトの検索を行う。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjects)(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount);
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE_PTR	IN/OUT	オブジェクトハンドルポインタ
	CK_ULONG	IN	最大オブジェクト数
	CK_ULONG_PTR	IN/OUT	オブジェクト数ポインタ

（１８） C_FindObjectsFinal

API 名	C_FindObjectsFinal
概要	オブジェクトの検索を終了する。

関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsFinal)(CK_SESSION_HANDLE hSession);
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

（１９） C_GetAttributeValue

API 名	C_GetAttributeValue
概要	オブジェクトの属性値を取得する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetAttributeValue)(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル
	CK_ATTRIBUTE_PTR	OUT	属性テーブルポインタ
	CK_ULONG	OUT	属性テーブル数

備考	以下の属性に対する値が取得可能である。	
	CKA_CLASS	CKO_CERTIFICATE または CKO_PRIVATE_KEY
	CKA_LABEL	表３オブジェクト情報一覧参照。
	CKA_VALUE	表３オブジェクト情報一覧参照。
	CKA_ISSUER	表３オブジェクト情報一覧参照。
	CKA_SERIAL_NUMBER	表３オブジェクト情報一覧参照。
	CKA_SUBJECT	表３オブジェクト情報一覧参照。
	CKA_ID	表３オブジェクト情報一覧参照。
	CKA_TOKEN	True
	CKA_PRIVATE	True
	CKA_CERTFICATE_TYPE	CKC_X_509
	CKA_MODULUS	表３オブジェクト情報一覧参照。
	CKA_MODULUS_BITS	表３オブジェクト情報一覧参照。
	CKA_PUBLIC_EXPONENT	表３オブジェクト情報一覧参照。
	CKA_KEY_TYPE	CKK_RSA
	CKA_SENSITIVE	True
	CKA_SIGN	True

表 3　オブジェクト属性一覧

#	オブジェクト	属性名	属性値
1	利用者鍵	CKA_LABEL	USERKEY
		CKA_ID	N の SHA1 ハッシュ
		CKA_PUBLIC_EXPONENT	利用者証明書から取得した値
		CKA_MODULUS	利用者証明書から取得した値
		CKA_MODULUS_BITS	利用者証明書から取得した値
2	利用者証明書	CKA_LABEL	USERCERT
		CKA_ID	N の SHA1 ハッシュ
		CKA_VALUE	証明書自体
		CKA_SUBJECT	利用者証明書から取得した値
		CKA_ISSUER	利用者証明書から取得した値
		CKA_SERIAL_NUMBER	利用者証明書から取得した値
3	都道府県知事の自己署名証明書	CKA_LABEL	CACERT
		CKA_ID	N の SHA1 ハッシュ
		CKA_VALUE	証明書自体
		CKA_SUBJECT	都道府県知事の自己署名証明書から取得した値

		CKA_ISSUER	都道府県知事の自己署名証明書から取得した値
		CKA_SERIAL_NUMBER	都道府県知事の自己署名証明書から取得した値

（２０） C_SignInit

API 名	C_SignInit		
概要	署名処理を初期化する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_SignInit)(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey);		
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_RSA_PKCS のみ指定可
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル

（２１） C_Sign

API 名	C_Sign
概要	データに署名を行う。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Sign)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen);

戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	データポインタ
	CK_ULONG	IN	データ長
	CK_BYTE_PTR	IN/OUT	署名データポインタ
	CK_ULONG_PTR	IN/OUT	署名データ長ポインタ

（２２） C_DigestInit

API 名	C_DigestInit		
概要	ダイジェスト作成を開始する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DigestInit)(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism);		
戻り値	CK_RV (CKR_OK: 成功 CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_SHA_1 のみ指定可

（２３） C_DigestUpdate

API 名	C_DigestUpdate		
概要	ダイジェストを作成する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DigestUpdate)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen);		
戻り値	CK_RV (CKR_OK: 成功 CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容

引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	ハッシュするデータポインタ
	CK_ULONG	IN	ハッシュするデータ長

（２４） C_DigestFinal

API 名	C_DigestFinal		
概要	ダイジェスト作成を終了する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DigestFinal)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN/OUT	ダイジェストデータポインタ
	CK_ULONG_PTR	IN/OUT	ダイジェストデータ長ポインタ

（２５） C_VerifyInit

API 名	C_VerifyInit		
概要	署名検証を開始する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_VerifyInit)(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism CK_OBJECT_HANDLE hKey);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_RSA_PKCS のみ指定可
	CK_OBJECT_HANDLE	IN	RSA 公開鍵オブジェクト

（２６） C_Verify

API 名	C_Verify
概要	署名値を検証する。

関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Verify)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen);
戻り値	CK_RV (CKR_OK: 成功 CKR_SIGNATURE_INVALID: 署名データ不正 CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	検証するデータポインタ
	CK_ULONG	IN	検証するデータ長
	CK_BYTE_PTR	IN	署名データポインタ
	CK_ULONG	IN	署名データ長

備考

pData と pSigneture の OID の関係は以下の通り。
署名データに OID をつける場合は、上位 UP にて検証するデータに OID を付加したデータを入力しなければならない。

		検証するデータ(pData)	
		OIDあり	OIDなし
署名データ (pSigneture)	OIDあり	OK	NG
	OIDなし	NG	OK

（２７） C_CreateObject

API 名	C_CreateObject
概要	公開鍵オブジェクトを作成する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_CreateObject)(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject);
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)

	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

	CK_ATTRIBUTE_PTR	IN	属性テーブルポインタ
	CK_ULONG	IN	属性テーブル数
	CK_OBJECT_HANDLE_PTR	IN/OUT	オブジェクトハンドルポインタ
備考	本 API では RSA 公開鍵セッションオブジェクトのみ作成できる。 以下の属性を pTemplate で指定する。 CK_OBJECT_CLASS: CKO_PUBLIC_KEY (※１) CKA_PUBLIC_EXPONENT CKA_MODULUS ※１ :CK_OBJECT_CLASS の値は CKO_PUBLIC_KEY 固定とする。 その他の属性は使用不可とする。		

（２８） C_DestroyObject

API 名	C_DestroyObject		
概要	公開鍵オブジェクトを破棄する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DestroyObject)(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject);		
戻り値	CK_RV (CKR_OK: 成功 CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル
備考	本 API では RSA 公開鍵セッションオブジェクトのみ破棄できる。		

第３節　構造体仕様

構造体は下記 URL にある PKCS#11 Ver2.0 の正式な資料を参照してください。

RSA Security 「http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/」

表４に本ライブラリで使用する構造体の一覧を示す。

表 ４　PKCS#11 構造体一覧

NO	構造体名	概要
1	CK_INFO CK_INFO_PTR	PKCS ライブラリ情報

2	CK_SLOT_ID CK_SLOT_ID_PTR	スロット ID
3	CK_SLOT_INFO CK_SLOT_INFO_PTR	スロット情報
4	CK_TOKEN_INFO CK_TOKEN_INFO_PTR	トークン情報
5	CK_SESSION_HANDLE CK_SESSION_HANDLE_PTR	セッションハンドル
6	CK_USER_TYPE	ユーザタイプ
7	CK_SESSION_INFO CK_SESSION_INFO_PTR	セッション情報
8	CK_OBJECT_HANDLE CK_OBJECT_HANDLE_PTR	オブジェクトハンドル
9	CK_OBJECT_CLASS CK_OBJECT_CLASS_PTR	オブジェクトクラス
10	CK_ATTRIBUTE CK_ATTRIBUTE_PTR	属性タイプ、値、長さを含む構造体
11	CK_MECHANISM_TYPE CK_MECHANISM_TYPE_PTR	メカニズムタイプ
12	CK_MECHANISM CK_MECHANISM_PTR	メカニズムタイプを含む、メカニズムを示す構造体
13	CK_MECHANISM_INFO CK_MECHANISM_INFO_PTR	メカニズム情報
14	CK_RV	ライブラリの戻り値
15	CK_NOTIFY	コールバック情報
16	CK_FUNCTION_LIST CK_FUNCTION_LIST_PTR CK_FUNCTION_LIST_PTR_PTR	PKCS ライブラリ関数

第４節　コーリングシーケンス

「第４章　第２節　実現可能な機能の一覧」を実現するためのコーリングシーケンスを以下に示す。上位アプリケーションは、このコーリングシーケンスに沿って実装すること。

（1） 初期処理

C_GetFunctionList	関数ポインタリストの取得 ppFunctionList: API アドレスポインタ
↓	
C_Initialize	PKCS#11 ライブラリの初期化 pReserved: NULL
↓	
C_GetSlotList	スロットリスト数の取得 tokenPresent: TRUE pSlotList: NULL pulCount: スロット数格納アドレス
↓	
C_GetSlotList	スロットリストの取得 tokenPresent: TRUE pSlotList: スロットリスト格納アドレス pulCount: スロット数格納アドレス
↓	
C_GetSlotInfo	スロット情報の取得 slotID: スロット ID pInfo: スロット情報格納アドレス
↓	
C_GetTokenInfo	トークン情報の取得 slotID: スロット ID pInfo: トークン情報格納アドレス
↓	
C_OpenSession	アプリケーションとトークン間のセッションの確立 slotID: スロット ID flags: 0 pApplication: NULL Notify: NULL phSession: セッションハンドル格納アドレス
↓	
C_Login	トークンへのログイン hSession: C_OpenSession で取得したハンドル

	userType: CKU_USER pPin: パスワード文字列 ulPinLen: パスワード文字列長

（2） 終了処理

C_Logout	トークンからのログアウト hSession: C_OpenSession で取得したハンドル
↓	
C_CloseSession もしくは、 C_CloseAllSessions	セッションの切断 C_CloseSession の場合 hSession: C_OpenSession で取得したハンドル C_CloseAllSessions の場合 slotID: C_OpenSession に指定したスロット ID
↓	
C_Finalize	PKCS#11 ライブラリの終了処理 pReseved: NULL

（3） 証明書取得処理

初期化処理	（1）初期処理参照
↓	
C_FindObjectsInit	証明書検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定 (1) type = CKA_CLASS value = CKO_CERTIFICATE (2) type = CKA_TOKEN value = TRUE ulCount: 設定する属性数(2)
↓	
C_FindObjects	証明書の検索 hSession: C_OpenSession で取得したハンドル phObject: オブジェクトハンドル格納アドレス ulMaxObjectCount: オブジェクトハンドル格納領域数(2) pulObjectCount: 発見したオブジェクト数格納アドレス
↓	
①（pulObjectCount – 1）回分ループ	

（カウンタを i、初期値を 0 とする）

↓

C_GetAttributeValue	証明書サイズの取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate: 以下の属性を指定 (1) type = CKA_LABEL（※１） value = NULL ulValueLen = サイズ格納領域アドレス (2) type = CKA_VALUE value = NULL ulValueLen = サイズ格納領域アドレス ulCount: 設定する属性数(2)

↓

C_GetAttributeValue	証明書の取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate: 以下の属性を指定 (1) type = CKA_LABEL（※１） value = ラベル格納アドレス ulValueLen =格納領域サイズ (2) type = CKA_VALUE value = 証明書格納アドレス ulValueLen = 格納領域サイズ ulCount: 設定する属性数(2)

↓

LABEL の値：求める証明書（※１）

≠ の場合 → カウンタ i を 1 加算して ①に戻る。

＝ の場合

↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

終了処理	（２）終了処理参照

（4） 署名生成処理*1

初期化処理	（1）初期処理参照

↓

C_FindObjectsInit	証明書検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定 (1) type = CKA_CLASS value = CKO_CERTIFICATE (2) type = CKA_TOKEN value = TRUE ulCount: 設定する属性数(2)

↓

C_FindObjects	証明書の検索 hSession: C_OpenSession で取得したハンドル phObject: オブジェクトハンドル格納アドレス ulMaxObjectCount: オブジェクトハンドル格納領域数(2) pulObjectCount: 発見したオブジェクト数格納アドレス

↓

①（pulObjectCount － 1）回分ループ
（カウンタを i、初期値を 0 とする）

↓

C_GetAttributeValue	証明書サイズの取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate:以下の属性を指定 (1) type = CKA_LABEL（※1） value = NULL ulValueLen = サイズ格納領域アドレス (2) type = CKA_VALUE value = NULL ulValueLen = サイズ格納領域アドレス ulCount:設定する属性数(2)

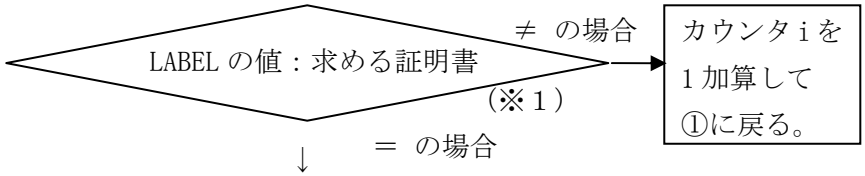
↓

C_GetAttributeValue	証明書の取得

*1 (4)署名生成処理のコーリングシーケンスを実行することで、ICカード内の利用者証明書、署名対象データに対するハッシュ値および署名値を取得することができる。

	hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate:以下の属性を指定 (1) type = CKA_LABEL (※1) value = ラベル格納アドレス ulValueLen =格納領域サイズ (2) type = CKA_VALUE value = 証明書格納アドレス ulValueLen = 格納領域サイズ ulCount:設定する属性数(2)

↓



↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

利用者証明書から公開鍵の E と N を取得

↓

C_FindObjectsInit	利用者鍵検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定する CKO_PRIVATE_KEY の指定は必須。他は任意。 (1) type = CKA_CLASS value = CKO_PRIVATE_KEY (2) type = CKA_TOKEN value = TRUE (3) type = CKA_MODULUS value = 公開鍵の N (4) type = CKA_PUBLIC_EXPONENT value = 公開鍵の E ulCount: 設定する属性数(4)

↓

C_FindObjects	利用者鍵の検索 hSession: C_OpenSession で取得したハンドル phObject: 利用者鍵オブジェクトハンドル ulMaxObjectCount: オブジェクトハンドル格納領域数

	(1) pulObjectCount: 発見したオブジェクト数格納アドレス

↓

C_FindObjectsFinal	利用者鍵の検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

C_DigestInit	ダイジェスト作成開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_SHA_1

↓

C_DigestUpdate	データをハッシュ hSession: C_OpenSession で取得したハンドル pPart: 署名対象データ ulPartLen: 署名対象データ長

↓

C_DigestFinal	ダイジェスト格納領域サイズ取得 hSession: C_OpenSession で取得したハンドル pDigest: NULL pulDigestLen: ダイジェストデータ長格納アドレス

↓

C_DigestFinal	ダイジェスト取得 hSession: C_OpenSession で取得したハンドル pDigest: ダイジェスト値格納アドレス pulDigestLen: ダイジェスト値長格納アドレス

↓

C_DigestFinal で取得したダイジェスト値（pDigest）を使用して DigestInfo を作成

↓

C_SignInit	署名処理の初期設定 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_RSA_PKCS hKey: C_FindObjects で取得したオブジェクトハンドル

↓

C_Sign	署名値長取得 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: NULL pulSignetureLen: 署名値長格納アドレス

↓

C_Sign	署名 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: 署名値格納アドレス pulSignetureLen: 署名値長格納アドレス

↓

終了処理	（2）終了処理参照

（5） 署名検証処理

初期化処理	（1）初期処理参照

↓

電子証明書から公開鍵の E と N を取得

↓

C_CreateObject	公開鍵オブジェクト作成 hSession: C_OpenSession で取得したハンドル pTemplate: 以下を指定する (1) type = CKA_CLASS value = CKO_PUBLIC_KEY (2) type = CKA_PUBLIC_EXPORNENT value = 公開鍵の E (3) type = CKA_MODULUS value = 公開鍵の N ulCount: 設定する属性数

↓

C_DigestInit	ダイジェスト作成開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_SHA_1

↓

C_DigestUpdate	データをハッシュ hSession: C_OpenSession で取得したハンドル pPart: 検証対象データ ulPartLen: 検証対象データ長

↓

C_DigestFinal	ダイジェスト格納領域サイズ取得 hSession: C_OpenSession で取得したハンドル

	pDigest: NULL pulDigestLen: ダイジェスト値長格納アドレス
↓	
C_DigestFinal	ダイジェスト取得 hSession: C_OpenSession で取得したハンドル pDigest: ダイジェスト値格納アドレス pulDigestLen: ダイジェスト値長格納アドレス
↓	
C_DigestFinal で取得したダイジェスト値（pDigest）を使用して DigestInfo を作成	
↓	
C_VerifyInit	データ検証開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_RSA_PKCS hKey: 公開鍵オブジェクトハンドル
↓	
C_Verify	データ検証 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: 署名値 pulSignetureLen: 署名値長
↓	
C_DestroyObject	公開鍵オブジェクト解放 hSession: C_OpenSession で取得したハンドル hObject: 対象オブジェクトハンドル
↓	
終了処理	（２）終了処理参照

（６） 繰り返し署名生成処理

「(４)署名生成処理」の網掛け部分を署名対象データの個数分だけ繰り返して呼び出す。

（７） 繰り返し署名検証処理

「(５)署名検証処理」の網掛け部分を検証対象データの個数分だけ繰り返して呼び出す。
（但し、すべての電子署名が同一の秘密鍵で生成された場合とする。）

第６章　その他

第１節　ライブラリのロード方法

汎用受付システム等の上位アプリケーションでは、ロードすべき PKCS#11 ライブラリのパス名を固定のファイル（ロード情報定義ファイル）から取得し、ロードする方式とする。

ロード情報定義ファイルのパス名およびファイル名は以下の通り。

```
C:¥Program Files¥Common Files¥e-gov_app¥load_path¥default.dat
```

（上記は OS のインストール先が C ドライブの場合）

ロード情報定義ファイル内のフォーマットは以下の通り。

```
項目=指定内容
```

- 項目と指定内容とは「=」でつなぎ、不要な空白等は含まない。
- 各項目について、内容は 1 行に記述し、「=」以降、改行文字までが有効な指定内容とする。

項目として設定する情報は以下の通り。

項目	指定内容
name	「会社名-識別情報」の形式。「jpki_appli-01」とする。
path	PKCS#11 ライブラリの格納先の絶対パス名を設定。 「C:¥Program Files¥JPKI¥JPKIPKCS11.dll」とする。

例)ロード情報定義ファイルの例

```
name=jpki_appli-01
path=C:¥Program Files¥JPKI¥JPKIPKCS11.dll
```

禁・無断転載

公的個人認証サービス

利用者クライアントソフト API 仕様書
【カード AP ライブラリ PKCS#11 編】

第１．１版

（注意事項）

※利用者クライアントソフトの著作権は、総務省が保有しており、国際著作権条約及び日本国の著作権関連法令によって保護されています。

※総務省は、利用者が利用者クライアントソフトを利用したことにより発生した利用者の損害及び利用者が第三者に与えた損害について、一切の責任を負いません。

※利用者クライアントソフトの利用に当たっては、次に掲げる行為を禁止します。

(1) 利用者クライアントソフトを電子申請・届出等の行政手続等以外の目的で利用すること。

(2) 利用者クライアントソフトに対し、総務省に許可なく改造等を行うこと。