付録 2-2
署名用トークンインターフェース仕様

PKCS#11 API Experiment Profile

- CONTENT -

- FIGURE CONTENT -

- TABLE CONTENT -

1 Introduction

1.1 Objective

This profile aims to be able to use the function of the signing and the verification together like using the product in each country when each country of the sphere of Asia uses one application service and to become it.

1.2 Overview

The specification to use each country's library based on PKCS#11 that is the international standard specification that U.S. RSA Laboratories settled on, and to use the function of the signing and the verification from a common application service together is settled on. The specification of the interface between the application and the PKCS#11 library, the function of PKCS#11, the execution sequence of the function, and the return code is settled on.

1.3 Review

This profile is corrected by the discussion based on trend and the situation of Asia.

1.4 Registered trademark

The company name, the product name and the brand name in this profile are the trademark of each company or the registered trademark.

1.5 Term definition

- PKCS#11 (Public Key Cryptograph Standard)

PKCS(Public Key Cryptograph Standard) is a standard concerning the public key cryptosystem technology that U.S. RSA Laboratories advocates. As for PKCS#11, it is said "Cryptographic Token Interface Standard", and is provided for API to operate the certificate call and store, signing and verifying using the private key and the public key, etc. to the token.

- token

The token indicates the generic name of the device that can be the carrying that can store the private key and the certificate for the owner identification. IC card is concretely given.

- RSA encryption algorithm

Public key cryptosystem algorithm that is invented by Mr. Ron Rivest, Mr. Adi

Shamir, and Mr. Leonard Adleman in 1977, and catches inventor's initial and was named. The factorization on prime numbers of a big number is assumed to be difficult of safety. It corresponds to a wide usage like not only encryption, decryption but also the signature, the verification, and the key distribution, etc.

- SHA-1

This is a correction version of Secure Hash Algorithm defined in FIPS 180. And this is a hush algorithm used when the e-signature is chiefly done

- public key

It is the key that pairs with the private key in the public key cryptosystem. It is open to the public as the certificate signed by trusted CA.
(Related item : private key)

- private key

It is the key that pairs with the public key in the public key cryptosystem. It is necessary to be managed by the end-entity strictly.
(Related item : public key)

2 The specification of Signing and verification

In this chapter, the specification of this profile is explained.

2.1 Application model

The application model assumed in this profile is shown at "Figure 2-1 Application Model". Here, Web Server is assumed to be the one to provide the application service, and Web Server is assumed to be used as common service in each country.

Otherwise, the client is assumed to be acquired the private key and public key.

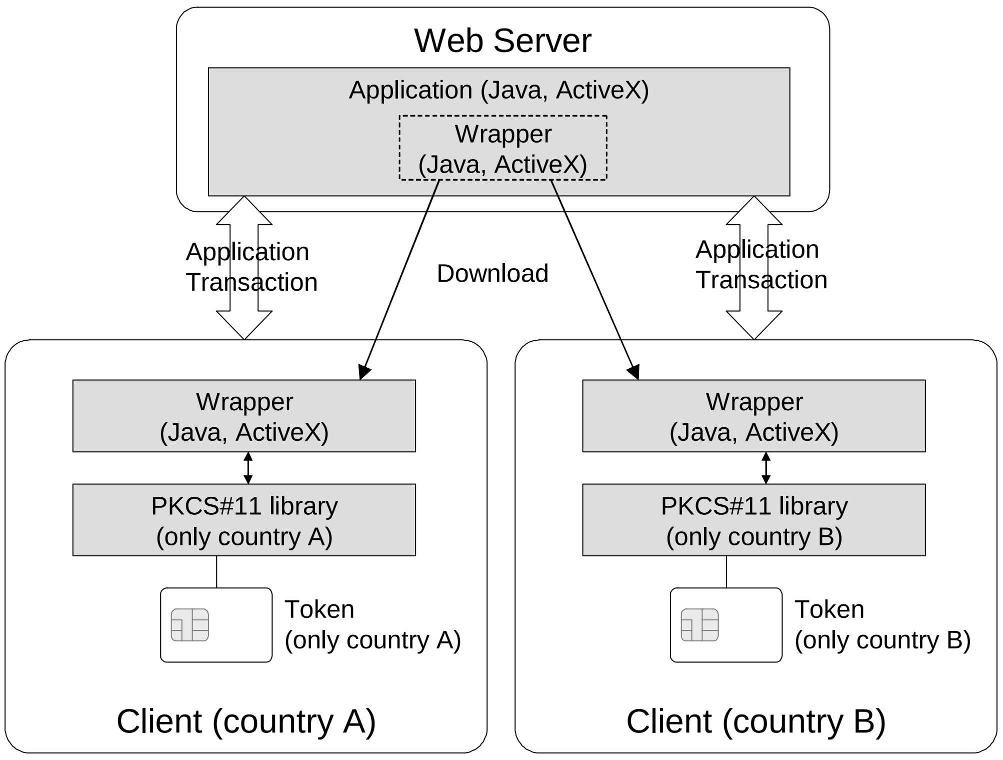

Figure 2-1 Application Model

In this model, at first, Web Server is providing an application service. In the client, it acquires the private key and public key under the PKCS#11 library management. The keys and PKCS#11 library is installed beforehand.

In the status that the client is accessing to Web Server, when the client processes a signing or verifying, the “wrapper” is downloaded from Web Server to the client, and the communication between the application and PKCS#11 library is enabled. The wrapper is Java applet or ActiveX.

The PKCS#11 library and token is used the one that it is possible to use it in each country. But the point of Wrapper that the application service provider offers it as a common module is important.

2.2 PKCS#11 Functions

In this term, it explains the functions that are used in all PKCS#11 functions.

2.2.1 PKCS#11 Version

In this profile, version 2.01 is required.

2.2.2 General Functions

In “General Functions”, it defines the functions used as a whole together. The functions in this group are shown as follows.

(1) C_Initialize

a) Format

```
CK_RV C_Initialize(CK_VOID_PTR pReserved);
```

b) Process

It initializes a PKCS#11 library.

c) Parameter

Parameter	Explanation
pReserved	NULL_PTR is specified.

(2) C_GetSlotList

a) Format

```
CK_RV C_GetSlotList(CK_BBOOL        tokenPresent,
                    CK_SLOT_ID_PTR pSlotList,
                    CK_ULONG_PTR    pulCount);
```

b) Process

The slot ID list or the number of slot is acquired.

c) Parameter

Parameter	Explanation
tokenPresent	The kind of slot is specified. TRUE : only the slot existing token FALSE : all slot
pSlotList	The area pointer that stores the slot ID lists in is specified.
pulCount	The area pointer that stores the number of slot.

(3) C_GetTokenInfo

a) Format

```
CK_RV C_GetTokenInfo(CK_SLOT_ID          slotID,
                     CK_TOKEN_INFO_PTR pInfo);
```

b) Process

It acquires the token information of the specified slot.

c) Parameter

Parameter	Explanation
slotID	The slot ID that corresponds to the token is specified.
pInfo	The pointer of ”CK_TOKEN_INFO” structure that stors a token information is specified.

(4) C_OpenSession

a) Format

```
CK_RV C_OpenSession(CK_SLOT_ID              slotID,
                    CK_FLAGS                flags,
                    CK_VOID_PTR             pApplication,
                    CK_NOTIFY               Notify,
                    CK_SESSION_HANDLE_PTR phSession);
```

b) Process

It opens the session between the process and token. And it acquires the session handle to identify the opened session.

c) Parameter

Parameter	Explanation
slotID	The slot ID that corresponds to the token is specified.
Flags	The session type is specified.
pApplication	NULL_PTR is specified.
Notify	NULL_PTR is specified.
phSession	The pointer of the area that store the session handle to identify the opened session.

(5) C_Login

a) Format

```
CK_RV C_Login(CK_SESSION_HANDLE   hSession,
              CK_USER_TYPE        userType,
              CK_CHAR_PTR         pPin,
              CK_ULONG            ulPinLen);
```

b) Process

It logs-in to the token.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
userType	User type is specified. CKU_SO or CKU_USER
pPin	The pointer to the PIN is specified.
ulPinLen	The length of PIN data is specified.

(6) C_Logout

a) Format

```
CK_RV C_Logout(CK_SESSION_HANDLE   hSession);
```

b) Process

It logs-out the user from the token.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.

(7) C_CloseSession

a) Format

```
CK_RV C_CloseSession(CK_SESSION_HANDLE hSession);
```

b) Process

The session is closed between token and process.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.

(8) C_Finalize

a) Format

```
CK_RV C_Finalize(CK_VOID_PTR pReserved);
```

b) Process

The end process of PKCS#11 library is done.

c) Parameter

Parameter	Explanation
pReserved	NULL_PTR is specified.

2.2.3 Management Functions

In “Management Functions”, it defines the functions that manage a private key, a public key and a token. The functions in this group are shown as follows.

(1) C_FindObjectsInit

a) Format

```
CK_RV C_FindObjectsInit(CK_SESSION_HANDLE hSession,
                        CK_ATTRIBUTE_PTR    pTemplate,
                        CK_ULONG            ulCount);
```

b) Process

The environment to search the objects that is same as the attribute specified in the template is initialized.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pTemplate	The pointer of the template that stores a serche conditions is specified.
ulCount	The number of the attribute that specified in the template is specified.

(2) C_FindObjects

a) Format

```
CK_RV C_FindObjects(CK_SESSION_HANDLE       hSession,
                    CK_OBJECT_HANDLE_PTR phObject,
                    CK_ULONG                ulMaxObjectCount,
                    CK_ULONG_PTR            pulObjectCount);
```

b) Process

The token objects and the session objects are searched by the condition set in C_FindObjectsInit function.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
phObject	The pointer of the area that is stored the object handle is specified.
ulMaxObjectCount	The max number of returning object handle is specified.
pulObjectCount	The pointer of the area that sotres the number of the object handle is specified.

(3) C_FindObjectsFinal

a) Format

```
CK_RV C_FindObjectsFinal(CK_SESSION_HANDLE hSession);
```

b) Process

The searching objects are finished.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.

(4) C_GetAttributeValue

a) Format

```
CK_RV C_GetAttributeValue(CK_SESSION_HANDLE hSession,
                          CK_OBJECT_HANDLE     hObject,
                          CK_ATTRIBUTE_PTR     pTemplate,
                          CK_ULONG             ulCount);
```

b) Process

The object attribute is acquired. Or the length of the attribute data is acquired.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
hObject	The object handle is specified.
pTemplate	The pointer of the template to receive the attribute is specified.
ulCount	The number of the template attribute is specified.

(5) C_CreateObject

a) Format

```
CK_RV C_CreateObject(CK_SESSION_HANDLE       hSession,
                     CK_ATTRIBUTE_PTR        pTemplate,
                     CK_ULONG                ulCount,
                     CK_OBJECT_HANDLE_PTR phObject);
```

b) Process

The object is added newly.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pTemplate	The pointer of the template set the object attribute is specified.
ulCount	The number of the attribute set in the template is specified.
phObject	The pointer of the area stored the new object handle is specified.

(6) C_DestroyObject

a) Format

```
CK_RV C_DestroyObject(CK_SESSION_HANDLE hSession,
                      CK_OBJECT_HANDLE   hObject);
```

b) Process

The object is annulled.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
hObject	The object handle is specified.

(7) C_GenerateKey (Optional)

a) Format

```
CK_RV C_GenerateKey(CK_SESSION_HANDLE   hSession,
                    CK_MECHANISM_PTR     pMechanism,
                    CK_ATTRIBUTE_PTR     pTemplate,
                    CK_ULONG             ulCount,
                    CK_OBJECT_HANDLE_PTR phKey);
```

b) Process

The common key is generated.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pMechanism	The pointer of generated common key mechanism is specified.
pTemplate	The pointer of generated common key attribute template is specified.
ulCount	The number of setting attribute in the template is specified.
phKey	The area pointer receiving the common key object handle is specified.

2.2.4 Signing Functions

In “Signing Functions”, it defines the signing functions and virification functions. The functions in this group is shown as follows.

(1) C_SignInit

a) Format

```
CK_RV C_SignInit(CK_SESSION_HANDLE hSession,
                 CK_MECHANISM_PTR   pMechanism,
                 CK_OBJECT_HANDLE   hKey);
```

b) Process

The signing process is initialized.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pMechanism	The pointer of signing mechanism is specified.
hKey	The object handle of the key is specified.

(2) C_Sign

a) Format

```
CK_RV C_Sign(CK_SESSION_HANDLE hSession,
             CK_BYTE_PTR        pData,
             CK_ULONG           ulDataLen,
             CK_BYTE_PTR        pSignature,
             CK_ULONG_PTR       pulSignatureLen);
```

b) Process

The single part data is signed.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pData	The pointer of the data area is specified.
ulDataLen	The data length is specified.
pSignature	The pointer of receiving a signed data area is specified.
pulSignatureLen	The pointer of receiving the signature length area is specified.

(3) C_VerifyInit

a) Format

```
CK_RV C_VerifyInit(CK_SESSION_HANDLE hSession,
                   CK_MECHANISM_PTR  pMechanism,
                   CK_OBJECT_HANDLE  hKey);
```

b) Process

The verification process is initialized.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pMechanism	The pointer of verification mechanism is specified.
hKey	The object handle of the key is specified.

(4) C_Verify

a) Format

```
CK_RV C_Verify(CK_SESSION_HANDLE hSession,
               CK_BYTE_PTR       pData,
               CK_ULONG          ulDataLen,
               CK_BYTE_PTR       pSignature,
               CK_ULONG          ulSignatureLen);
```

b) Process

The single part data is verified.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pData	The pointer of the data is specified.
ulDataLen	The data length is specified.
pSignature	The pointer of signature is specified.
ulSignatureLen	The length of signature is specified.

2.2.5 Encryption Functions (Optional)

In "Encryption Functions", it defines the encryption functions and decryption functions. The functions in this group are shown as follows.

The functions in this group are optional.

(1) C_EncryptInit

a) Format

```
CK_RV C_EncryptInit(CK_SESSION_HANDLE hSession,
                    CK_MECHANISM_PTR   pMechanism,
                    CK_OBJECT_HANDLE   hKey);
```

b) Process

The encryption process is initialized.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pMechanism	The pointer of encryption mechanism is specified.
hKey	The object handle of the key is specified.

(2) C_Encrypt

a) Format

```
CK_RV C_Encrypt(CK_SESSION_HANDLE hSession,
                CK_BYTE_PTR        pData,
                CK_ULONG           ulDataLen,
                CK_BYTE_PTR        pEncryptedData,
                CK_ULONG_PTR       pulEncryptedDataLen);
```

b) Process

The single part data is encrypted.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pData	The pointer of normal data is specified.
ulDataLen	The length of normal data is specified.
pEncryptedData	The area pointer of receiving encrypted data is specified.
pulEncryptedDataLen	The area pointer of receiving encrypted data length is specified.

(3) C_DecryptInit

a) Format

```
CK_RV C_DecryptInit(CK_SESSION_HANDLE hSession,
                    CK_MECHANISM_PTR   pMechanism,
                    CK_OBJECT_HANDLE   hKey);
```

b) Process

The decryption process is initialized.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pMechanism	The pointer of verification mechanism is specified.
hKey	The object handle of the key is specified.

(4) C_Decrypt

a) Format

```
CK_RV C_Decrypt(CK_SESSION_HANDLE hSession,
                CK_BYTE_PTR        pData,
                CK_ULONG           ulDataLen,
                CK_BYTE_PTR        pDecryptedData,
                CK_ULONG_PTR       pulDecryptedDataLen);
```

b) Process

The single part data is decrypted.

c) Parameter

Parameter	Explanation
hSession	The session handle that is acquired in C_OpenSession is specified.
pData	The pointer of encrypted data is specified.
ulDataLen	The length of encrypted data is specified.
pDecryptedData	The area pointer of receiving decrypted data is specified.
pulDecryptedDataLen	The pointer of receiving decrypted data length is specified.

2.3 Wrapper Functions

In this term, the method that the wrapper is implemented the functions shown at “2.2 PKCS#11 Functions” is explained.

The application developer may mount the wrapper without doing as an independent module as one function of the application.

2.3.1 PKCS#11 Functions Sequences

In the wrapper, it starts making the function shown at “2.2 PKCS#11 Functions” a group each processing, and it defines the sequences. In the application, communication to the PKCS#11 library is enabled by calling these sequences.

The sequences are shown as follows.

(1) Initialize Sequence

- C_Initialize
- C_GetSlotList
- C_GetTokenInfo
- C_OpenSession
- C_Login

(2) Logout Sequence

- C_Logout
- C_CloseSession
- C_Finalize

(3) Get Certificate Sequence

- C_FindObjectsInit
- C_FindObjects
- C_FindObjectsFinal
- C_GetAttributeValue

(4) Sign Sequence

- C_SignInit
- C_Sign

(5) Verify Sequence

- C_CreateObject
- C_VerifyInit
- C_Verify

(6) Delete Object Sequence

- C_DestroyObject

(7) Common Key Generate Sequence (Optional)

- C_GenerateKey
- C_GetAttributeValue

(8) Encryption Sequence (Optional)

- C_EncryptInit
- C_Encrypt

(9) Decryption Sequence (Optional)

- C_DecryptInit
- C_Decrypt

2.3.2 PKCS#11 Return Codes

About the return codes of each function in the sequences that are shown at "2.3.1 PKCS#11 Functions Sequences", the return codes that are contained in this profile are shown at "Table 2-1 Return Code List".

The adoption of these return codes is left to the application developer's judgment.

Table 2-1 Return Code List

No.	Function	Return Code	Notes
1	C_GetFunctionList	CKR_OK	
2	C_Initialize	CKR_OK	
3	C_Finalize	CKR_OK	
4	C_GetTokenInfo	CKR_OK	
5		CKR_DEVICE_REMOVED	
6	C_GetSlotList	CKR_OK	
7	C_OpenSession	CKR_OK	
8		CKR_DEVICE_REMOVED	
9		CKR_TOKEN_NOT_PRESENT	
10		CKR_SESSION_HANDLE_INVALID	
11		CKR_SLOT_ID_INVALID	
12	C_CloseSession	CKR_OK	
13		CKR_TOKEN_NOT_PRESENT	
14		CKR_SESSION_HANDLE_INVALID	
15		CKR_DEVICE_REMOVED	
16	C_Login	CKR_OK	
17		CKR_DEVICE_REMOVED	
18		CKR_TOKEN_NOT_PRESENT	
19		CKR_SESSION_HANDLE_INVALID	
20		CKR_PIN_INCORRECT	
21		CKR_PIN_INVALID	
22		CKR_PIN_LEN_RANGE	
23		CKR_PIN_LOCKED	
24		CKR_USER_ALREADY_LOGGED_IN	
25	C_Logout	CKR_OK	
26		CKR_DEVICE_REMOVED	
27		CKR_TOKEN_NOT_PRESENT	
28		CKR_SESSION_HANDLE_INVALID	
29		CKR_USER_NOT_LOGGED_IN	
30	C_FindObjectsInit	CKR_OK	
31		CKR_TOKEN_NOT_PRESENT	
32		CKR_SESSION_HANDLE_INVALID	
33		CKR_DEVICE_REMOVED	
34	C_FindObjects	CKR_OK	
35		CKR_TOKEN_NOT_PRESENT	
36		CKR_SESSION_HANDLE_INVALID	
37		CKR_DEVICE_REMOVED	
38	C_FindObjectsFinal	CKR_OK	
39		CKR_TOKEN_NOT_PRESENT	
40		CKR_SESSION_HANDLE_INVALID	

No.	Function	Return Code	Notes
41		CKR_DEVICE_REMOVED	
42	C_CreateObject	CKR_OK	
43		CKR_DEVICE_REMOVED	
44		CKR_TOKEN_NOT_PRESENT	
45		CKR_SESSION_HANDLE_INVALID	
46		CKR_USER_NOT_LOGGED_IN	
47	C_DestroyObject	CKR_OK	
48		CKR_TOKEN_NOT_PRESENT	
49		CKR_SESSION_HANDLE_INVALID	
50		CKR_DEVICE_REMOVED	
51	C_GetAttributeValue	CKR_OK	
52		CKR_TOKEN_NOT_PRESENT	
53		CKR_SESSION_HANDLE_INVALID	
54		CKR_DEVICE_REMOVED	
55	C_SignInit	CKR_OK	
56		CKR_TOKEN_NOT_PRESENT	
57		CKR_SESSION_HANDLE_INVALID	
58		CKR_DEVICE_REMOVED	
59		CKR_USER_NOT_LOGGED_IN (optional)	
60	C_Sign	CKR_OK	
61		CKR_DATA_INVALID	
62		CKR_DATA_LEN_RANGE	
63		CKR_TOKEN_NOT_PRESENT	
64		CKR_SESSION_HANDLE_INVALID	
65		CKR_DEVICE_REMOVED	
66	C_VerifyInit	CKR_OK	
67		CKR_TOKEN_NOT_PRESENT	
68		CKR_SESSION_HANDLE_INVALID	
69		CKR_DEVICE_REMOVED	
70		CKR_USER_NOT_LOGGED_IN	
71	C_Verify	CKR_OK	
72		CKR_DATA_INVALID	
73		CKR_DATA_LEN_RANGE	
74		CKR_TOKEN_NOT_PRESENT	
75		CKR_SESSION_HANDLE_INVALID	
76		CKR_SIGNATURE_INVALID	
77		CKR_DEVICE_REMOVED	
78		CKR_SIGNATURE_LEN_RANGE	
79	C_EncryptInit	CKR_OK	
80		CKR_TOKEN_NOT_PRESENT	
81		CKR_SESSION_HANDLE_INVALID	
82		CKR_DEVICE_REMOVED	
83	C_Encrypt	CKR_OK	
84		CKR_TOKEN_NOT_PRESENT	
85		CKR_SESSION_HANDLE_INVALID	
86		CKR_DEVICE_REMOVED	
87	C_DecryptInit	CKR_OK	
88		CKR_TOKEN_NOT_PRESENT	
89		CKR_SESSION_HANDLE_INVALID	
90		CKR_DEVICE_REMOVED	
91	C_Decrypt	CKR_OK	
92		CKR_TOKEN_NOT_PRESENT	
93		CKR_SESSION_HANDLE_INVALID	

No.	Function	Return Code	Notes
94		CKR_DEVICE_REMOVED	
95		CKR_ENCRYPTED_DATA_LEN_RANGE	

2.3.3 PKCS#11 library loading

If the wrapper is used commonly in each country, there is one problem. There is a difference between the file name of PKCS#11 library used in each country. So, when the wrapper is downloaded in the client, the wrapper can not know the file name.

To resolve this problem, the method that the file name is written in initialize file is adopted. Detail is shown at “Figure 2-2 File Name Resolving”.

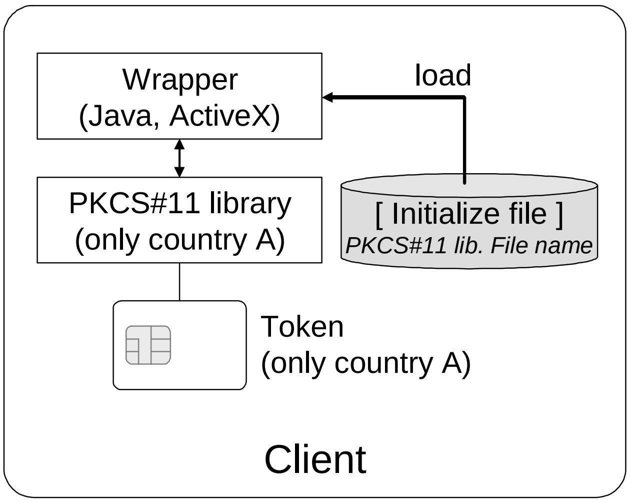

Figure 2-2 File Name Resolving

The specification of this initialize file is as follows.

(1) File Name

The file name is “pkcs11.ini”.

(2) Contents

The contents of “pkcs11.ini” is as follows.

```
1 : [PKCS11.Driver.Name]
2 : F3EZscl2.dll
```

In the 1st line, it declares that the file name of PKCS#11 library is written in this.

In the 2nd line, it writes the file name of PKCS#11 library of each country.

(3) Saved place

The saved place of "pkcs11.ini" is at the system folder in Windows installed drive and folder. Moreover, the file (DLL etc.) that is the realities of PKCS#11 library is saved the folder as same as "pkcs11.ini".

Example)	Windows installed drive	: C drive
	Windows installed folder	: winnt
	System folder of Windows	: system32

In this case, the place of pkcs11.ini is C:¥winnt¥system32¥.

2.4 Application Interface

When the application calls PKCS#11 functions, concretely, the application calls the sequences that are defined at "2.3.1 PKCS#11 Functions Sequences" to the wrapper. Details are shown at "Figure 2-3 PKCS#11 Function Call from Application".

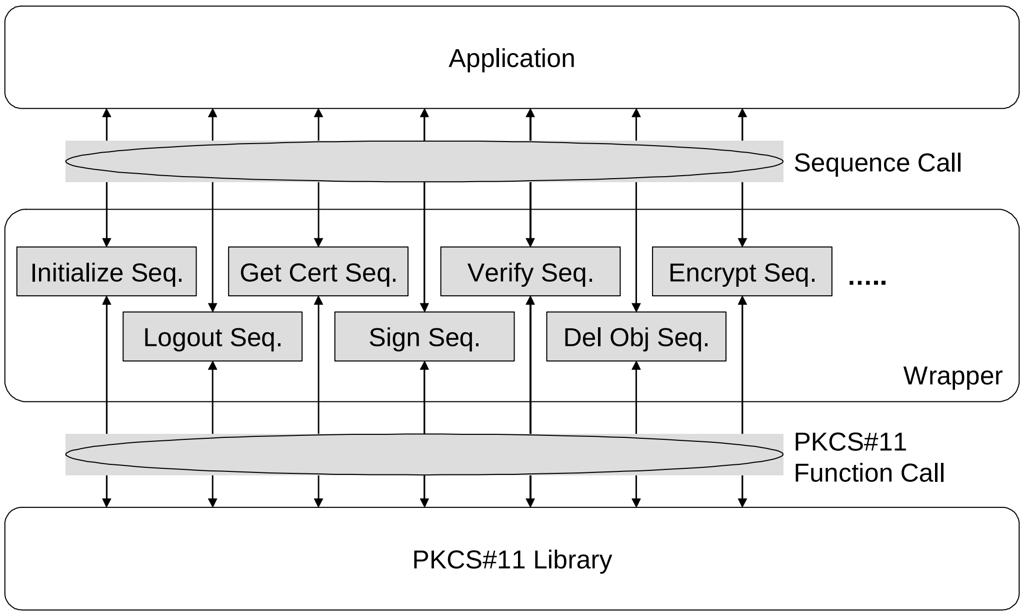

Figure 2-3 PKCS#11 Function Call from Application

2.5 PKCS#11 Other Factors

In this term, the other factors are explained.

2.5.1 Algorithm

The signing algorithm used in C_Sign function and C_Verify function is RSA encryption or SHA-1 with RSA encryption.

2.5.2 Key Length

The key length of public key and private key are 1024 bits.

2.5.3 Signing Mechanism

The mechanism specified in C_SignInit function and C_VerifyInit function is CKM_RSA_PKCS or CKM_SHA1_RSA_PKCS.