

Mobile Certificate Application Specification (C Version 1.0)

Copyright: All right reserved SecuTech Solution, inc.

Table of Content

1. Relative Convention	1
1.1 Character Coding	1
1.2 Return Value	1
1.3 Public Key Format	1
1.4 Hash Algorithm Identification	1
1.5 Ease of Use Convention	1
1.6 Third Party Encrypt Library.....	2
1.7 Relative Development.....	2
1.8 Development Tools	2
1.9 Type Definition.....	2
2 Interface Definition	2
2.1 Device Connection Status.....	2
2.2 Generate a Key-pair.....	3
2.3 Import Public Key.....	3
2.4 Digital Sign (including including explicit key signature, ordinary signature, SSL handling, and P10 signature).....	3
2.5 Decrypt Digital Envelope	4
2.6 Change Password	4
2.7 Get DER Code of Public Key Certificate	5
2.8 Get the Remaining Number of Password Attempts	5
2.9 Get device Serial Number (the unique device serial number)	5
3 Error Code Table.....	6

1. Relative Convention

1.1 Character Coding

All string parameters are using utf-8 coding standard.

1.2 Return Value

All return values from functions are long. 0 means success, and <0 indicates an error code.

1.3 Public Key Format

pPublicKey is compatible with RFC 3447 specification.

```
RSAPublicKey ::= SEQUENCE {  
    modulus             INTEGER  -- n  
    publicExponent      INTEGER  -- e  
}
```

1.4 Hash Algorithm Identification

pHashOID is the identification string of each hash algorithm, and the specific algorithm ID are as the following table.

ID	Specification
1.3.14.3.2.26	SHA1 Algorithm
2.16.840.1.101.3.4.2.1	SHA256 Algorithm

1.5 Ease of Use Convention

As the speed of reading a certificate is low, an initiation mechanism shall be established, that is after the content of a public key is read at the first time of inserting a key, the public key and the information of the corresponding private key, such as container name, container ID and etc, shall be stored together in a configure file (the configure file will be created automatically: such as

“company name.config”) to ensure that at the next time, the public key and private key-pair can be established by just reading the private ID in the key. Through this method, the speed of certificate reading will be improved, and meanwhile the private key can be found very fast when signing or decrypting digital envelope.

1.6 Third Party Encrypt Library

If using encrypt library, please use libCrypto.so in Android system. When the library is called, it should support Android v2.1 and later.

1.7 Relative Development

1. Functions, variables and etc. shall comply with C standard.
2. Interfaces with binary returns, such as SignData, shall be called twice, that is, at the first time, get the length of data, after applying for memory space externally, return specific data at the second time.
3. All passed char* type parameters shall be ended with '\0', and the length can be calculated directly

1.8 Development Tools

Use NDK built-in tools version 1.6 or later as the compiler tools.

1.9 Type Definition

```
Typedef      unsigned char   byte;  
Typedef      byte            *pbyte;
```

2 Interface Definition

2.1 Device Connection Status

```
long  IsConnected (bool* bConnected);
```

Parameters:

[out] bConnected Device connection status

Returns:

S_OK Success
<0 Fail, return error code.

2.2 Generate a Key-pair

```
long CreateRSAKey(char* pPassword, long lPublicKeyLen, pbyteppPublicKey);
```

Parameters:

[in] pPassword Device Password
[in] lPublicKeyLen Specify public key length: 1024 or 2048
[out] ppPublicKey Public key of the generated key-pair (format comply with the coding specification described in this document)

Returns:

S_OK Success
<0 Fail, return error code.

2.3 Import Public Key Certificate

```
long ImportX509Certificate(pbyte pCertificate, long lCertificateLen);
```

Parameters:

[in] pCertificateDer Public key encoding X.509
[in] lCertificateLen Public key length

Returns:

S_OK Success
<0 Fail, return error code.

2.4 Digital Sign (including including explicit key signature, ordinary signature, SSL handling, and P10 signature)

```
long SignData(
    pbyte pCertificate, long lCertificateLen, char* pPassword,
    pbyte pSrcData, long lSrcDataLen, char* pHashOID, pbyte
    ppOutData, long* lpOutDataLen
);
```

Parameters:

[in] pCertificate Public key certificate corresponding to private key, and if RSA key pairs are just generated, use the public key value returned from

3

createRSAkey function.

[in]	lCertificateLen	Public key length
[in]	pPassword	Device password
[in]	pSrcData	Source data to be signed
[in]	pSrcDataLen	length of source data to be signed
[in]	pHashOIDHash	Algorithm ID
[out]	ppOutData	Signed data (PKCS1 format)
[out]	lpOutDataLen	length of signed data (PKCS1 format)

Returns:

S_OK	Success
<0	Fail, return error code.

2.5 Decrypt Digital Envelope

```
long DecryptEnvelopeData(
    pbyte pCertificate, long lCertificateLen, char
    * pPassword, pbyte pInData, long lInDataLen,
    pbyte ppOutData, long* lOutDataLen
);
```

Parameters:

[in]	pCertificate	Public key certificate corresponding to the private key, which is the decrypting certificate
[in]	lCertificateLen	Public key length
[in]	pPassword	Device password
[in]	pInData	Cipher text of digital envelope(include PKCS7 format and CMS format)
[in]	lInDataLen	length of cipher text of digital envelope
[out]	ppOutData	Original data of digital envelope
[out]	lOutDataLen	length of original data of digital envelope

Returns:

S_OK	Success
<0	Fail, return error code.

2.6 Change Password

```
longChangePassword(
    pbyte pCertificate, long lCertificateLen, char*
    pOldPwd, char* pNewPwd
);
```

Parameters:

[in]	pCertificate	Public key certificate, and if it's device password, ignore this parameter.
------	--------------	---

[in]	lCertificateLen	Public key length, and if it's device password, ignore this parameter
[in]	pOldPwd	Old password
[in]	pNewPwd	New Password

Returns:

S_OK	Success
<0	Fail, return error code.

2.7 Get DER Code of Public Key Certificate

```
longGetX509Certificates(int* ipCertificates, long** lppCertificateLen,
                        pbyte* ppCertificates);
```

Parameters:

[in,out]	ipCertificates	Number of public key(the return value of the function that is called at the first time)
[in,out]	lppCertificateLen	Length of every public key(the return value of function that is called at the second time)
[out]	ppCertificates	The two-dimension array of public key DER encoded(the return value of the function that is called at the third time)

Returns:

S_OK	Success
<0	Fail, return error code.

2.8 Get the Remaining Number of Password Attempts

```
long GetPwdCanRetries(pbyte pCertificate, long lCertificateLen,
                      int*ipRemaining);
```

Parameters:

[in]	pCertificate	Public key certificate, and if it's device password, ignore this parameter.
[in]	lCertificateLen	Public key length, and if it's device password, ignore this parameter
[out]	ipRemaining	Number of password attempts

Returns:

S_OK	Success
<0	Fail, return error code.

2.9 Get device Serial Number (the unique device serial number)

```
long GetDeviceSerialNumber( char* ppDeviceSN, long* lpDeviceSNLen);
```

Parameters:

[out] ppDeviceSN Device serial number
[out] lpDeviceSNLen length of device serial number

Returns:

S_OK Success
<0 Fail, return error code.

3 Error Code Table

Error Code	Macro definition	Remark
0	S_OK	Success
-1	S_DEVICE_FAILURE	No connection
-2	S_INVALID_PASSWORD	Verify password failed
-3	S_DEVICE_LOCK	Device Locked
-4	S_NOT_ENOUGH_MEMORY	No enough spaces of Memory
-5	S_INVALID_CERTIFICATE	Invalid format of Certificate
-6	S_INVALID_ALGORITHM	Invalid algorithm
-7	S_INVALID_PARAMETER	Invalid parameter
-8	S_UNKOWN_ERROR	Unknown error
-9	S_USER_CANCEL	Canceled by user