PKCS #11: Cryptographic Token Interface Standard (Cryptoki)

An RSA Laboratories Seminar

Revised September 12, 1995

Copyright © 1995 RSA Laboratories, a division of RSA Data Security, Inc.
All rights reserved. 901-903008-132-000-000

Introduction

- Portable devices ideal for public-key cryptography
  - store private keys securely
  - cryptographic operations performed by device
- Standard programming interface increasingly important
- PKCS #11 (Cryptoki) offered as candidate

Status

- Announced in January 1994
- First draft in February
- Review meeting May 19–20
- Second draft released October 19
- Third draft released March 22, 1995
- Pre-release of final document April 22, 1995
- Final document released April 28, 1995

Design goals

1. Technology independence

   – any kind of cryptographic device

   – any operating system

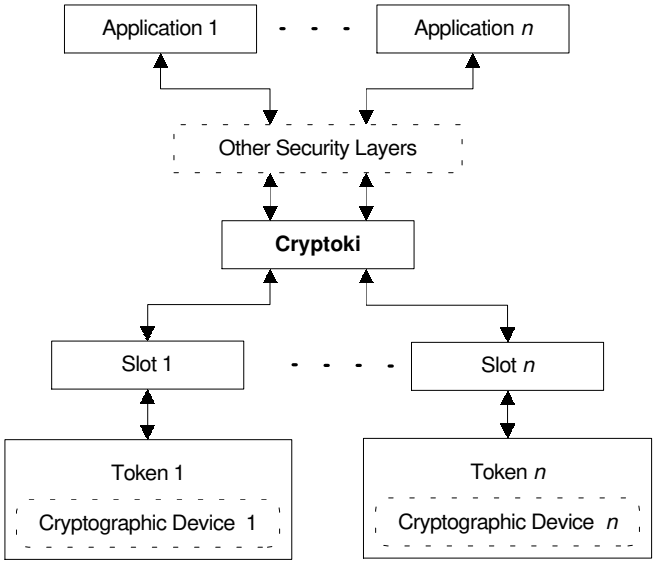
2. Resource sharing

   – more than one application per device, or device per application

General model

- Cryptoki is a programming interface that gives a common logical view to all kinds of cryptographic device
  - a cryptographic *token*
  - "software" devices also supported
- A given Cryptoki implementation supports certain classes of device
  - initially, each class expected to have its own Cryptoki implementation
  - eventually, a single implementation for all classes?

General model (cont'd)



- "Other Security Layers" might include:
  - Generic Security Services Application Programming Interface (RFCs 1508 and 1509)
  - Generic Cryptographic Service API (X/Open)
  - Cryptographic API (CAPI) (Microsoft)

Programming model

- Cryptoki maintains a list of slots that interface to tokens, and a list of tokens “in the system”
- Application opens a session with a token, and a user “logs in”
- Token stores collection of objects
  - Application data, certificates, and keys

Tokens and slots

- Application can obtain information about each slot, such as:
  - description (e.g., “PCMCIA”)
  - if the device is removable
  - if the slot is a “hardware” or “software” slot
  - if a token is present

Tokens and slots (cont'd)

- Application can obtain a list of tokens in the system and information about each one, such as:
  - a label
  - description (manufacturer, model)
  - serial number
  - status (read-only, random number generator)
  - maximum of sessions
  - maximum and minimum length of PIN
  - amount of total and free memory on the token

Sessions

- Application opens a session with a token
  - read-only session
  - read/write session
- For greater access, application user logs in
  - requires PIN
  - Security Officer or normal user access

Objects

- Classes
  - Application data objects hold information defined by an application
  - Certificate objects hold public-key certificates
  - Key objects hold public, private, and secret keys
- Lifetime
  - *Token* objects exist on token and remain after session is closed
  - *Session* objects exist only for the duration of the session

Objects

- Visibility
  - Public objects are visible to all sessions
  - Private objects are visible only after a user has logged in
- Objects consist of *attributes*
- An application can create or destroy objects, and manipulate attributes
- An application can also search for objects based on attributes
- Attributes supported depends on application profile

Object types

- Application data
- Certificate
- Public key
  - RSA
  - DSA
  - Diffie-Hellman
- Private key
  - RSA
  - DSA
  - Diffie-Hellman
- Secret key
  - RC2
  - RC4
  - DES (single, double, or triple length)

Cryptographic operations

- An application can perform cryptographic operations on objects and data supplied by the application
  - operation may run in parallel with application
- Cryptographic operations supported depends on mechanisms supported by device

Functions

- **`C_Initialize`**: initializes Cryptoki
- **`C_GetInfo`**: obtains general information about Cryptoki
- **`C_GetSlotList`**: obtains a list of slots in the system
- **`C_GetSlotInfo`**: obtains information about a particular slot
- **`C_GetTokenInfo`**: obtains information about a particular token
- **`C_GetMechanismList`**: obtains a list of mechanisms supported by a token
- **`C_GetMechanismInfo`**: obtains information about a particular mechanism supported by a token

Functions (cont'd)

- **`C_InitToken`**: initializes a token
- **`C_InitPIN`**: initializes the normal user's PIN
- **`C_SetPIN`**: modifies the PIN of the current user
- **`C_OpenSession`**: opens a connection or "session" between an application and a particular token
- **`C_CloseSession`**: closes a session
- **`C_CloseAllSessions`**: closes all sessions with a token
- **`C_GetSessionInfo`**: obtains information about the session

Functions (cont'd)

- **`C_Login`**: logs into a token
- **`C_Logout`**: logs out from a token
- **`C_CreateObject`**: creates an object
- **`C_CopyObject`**: creates a copy of an object
- **`C_DestroyObject`**: destroys an object
- **`C_GetObjectSize`**: obtains the size of an object in bytes
- **`C_GetAttributeValue`**: obtains an attribute value of an object
- **`C_SetAttributeValue`**: modifies an attribute value of an object
- **`C_FindObjectsInit`**: initializes an object search operation
- **`C_FindObjects`**: continues an object search operation

Functions (cont'd)

- **`C_EncryptInit`**: initializes an encryption operation
- **`C_Encrypt`**: encrypts single-part data
- **`C_EncryptUpdate`**: continues a multiple-part encryption operation
- **`C_EncryptFinal`**: finishes a multiple-part encryption operation

Functions (cont'd)

- **`C_DecryptInit`**: initializes a decryption operation
- **`C_Decrypt`**: decrypts single-part encrypted data
- **`C_DecryptUpdate`**: continues a multiple-part decryption operation
- **`C_DecryptFinal`**: finishes a multiple-part decryption operation

Functions (cont'd)

- **`C_DigestInit`**: initializes a message-digesting operation
- **`C_Digest`**: digests single-part data
- **`C_DigestUpdate`**: continues a multiple-part digesting operation
- **`C_DigestFinal`**: finishes a multiple-part digesting operation

Functions (cont'd)

- **`C_SignInit`**: initializes a signature operation
- **`C_Sign`**: signs single-part data
- **`C_SignUpdate`**: continues a multiple-part signature operation
- **`C_SignFinal`**: finishes a multiple-part signature operation
- **`C_SignRecoverInit`**: initializes a signature operation, where the data can be recovered from the signature
- **`C_SignRecover`**: signs single-part data, where the data can be recovered from the signature

Functions (cont'd)

- **`C_VerifyInit`**: initializes a verification operation
- **`C_Verify`**: verifies a signature on single-part data
- **`C_VerifyUpdate`**: continues a multiple-part verification operation
- **`C_VerifyFinal`**: finishes a multiple-part verification operation
- **`C_VerifyRecoverInit`**: initializes a verification operation where the data is recovered from the signature
- **`C_VerifyRecover`**: verifies a signature on single-part data, where the data is recovered from the signature

Functions (cont'd)

- **`C_GenerateKey`**: generates a secret key
- **`C_GenerateKeyPair`**: generates a public-key/private-key pair
- **`C_WrapKey`**: wraps (encrypts) a key
- **`C_UnwrapKey`**: unwraps (decrypts) a key
- **`C_DeriveKey`**: derives a key from a base key
- **`C_SeedRandom`**: mixes in additional seed material to the random number generator
- **`C_GenerateRandom`**: generates random data

Functions (cont'd)

- **`C_GetFunctionStatus`**: obtains updated status of a function running in parallel with the application
- **`C_CancelFunction`**: cancels a function running in parallel with the application
- **`Notify`**: callback function to process notifications from Cryptoki

Example: Signing a message

1. Find token
   - **`C_GetSlotList`**, **`C_GetTokenInfo`** (one or more times)
2. Open session, and log in
   - **`C_OpenSession`**, **`C_Login`**
3. Find key that supports signatures
   - **`C_FindObjects`**
4. Run signature functions
   - **`C_SignInit, C_Sign`**
5. Log out and close session
   - **`C_Logout`**, **`C_CloseSession`**

Example application profiles

- Privacy-Enhanced Mail (PEM)
  - RSA, DES, MD2, MD5
- Government authentication-only
  - DSA, SHA
- Cellular Digital Packet Data (CDPD)
  - Diffie-Hellman, RC4
- More to be defined

Conclusions

- Cryptoki is a standard programming interface for portable cryptographic devices
- More application profiles to be defined
- First libraries demonstrated at 1995 RSA Data Security conference, more to come
- RSA has a software-only implementation of Cryptoki called “SofToken”
- RSA also has a Cryptoki test program for Windows 3.1x Cryptoki DLL’s

Obtaining copies

- E-mail responder: **`pkcs@rsa.com`**
- FTP: **`rsa.com`**, **`pub/pkcs/pkcs-11`** directory
- Mailing list: **`pkcs-11-dev@rsa.com`**
  - to subscribe: **`pkcs-11-dev-request`**