ICカードなどのハードウェアトークンAPI

大日本印刷株式会社
ビジネスフォーム事業部
ビジネスソリューション開発本部

半田　富己男
Handa-F@mail.dnp.co.jp

2004 **年** 8**月** 26**日**

ICカードなどのハードウェアトークンAPI

- **ハードウェアトークン とは**
- PKI**アプリケーション**とIC**カード**
- PKCS#11
- CSP (Cryptographic Service Provider)
- PKCS#11**と**CSP**の相互運用**
- PKCS#15
- GSC-IS**概観**

ハードウェアトークンとは

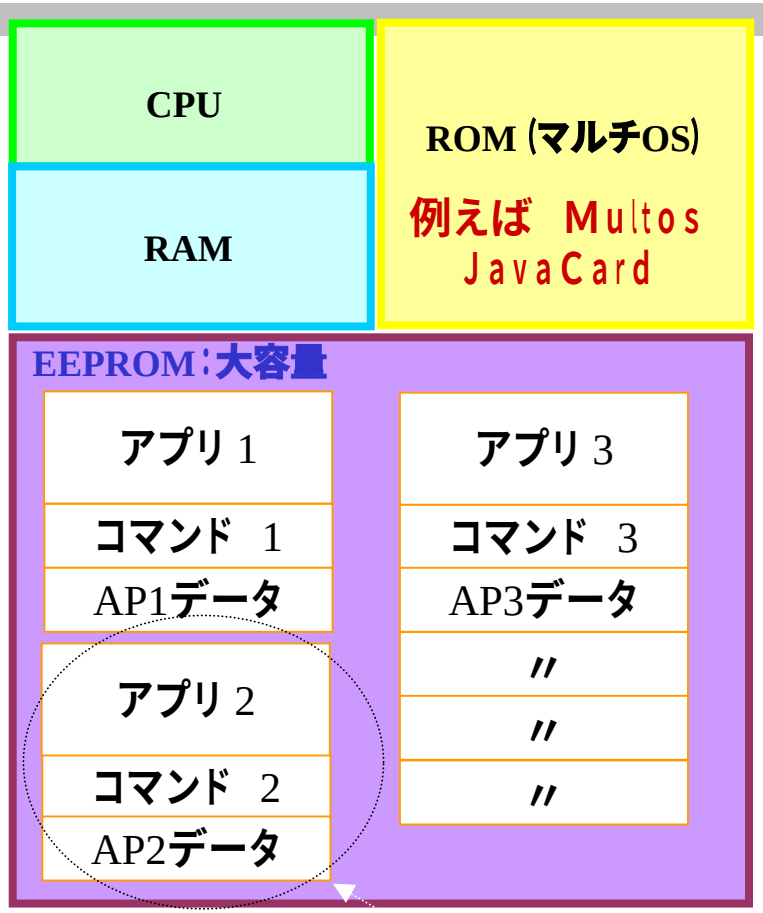
- **トークン** (identity token)
  - **トークン所持者の身元認証**(identity authentication)**に利用する情報を格納し、持ち運ぶことが可能な装置。**
- **ハードウェアトークンの例**
  - **IC**カード (smart card)
  - USB**キー**

- **トークン所持者の身元認証に利用する情報**
  - **公開鍵証明書**
  - **プライベート鍵**
    - **プライベート鍵はトークン所持者以外の第三者にアクセスさせない。**
    - **プライベート鍵がハードウェアトークンから外に出ない。**

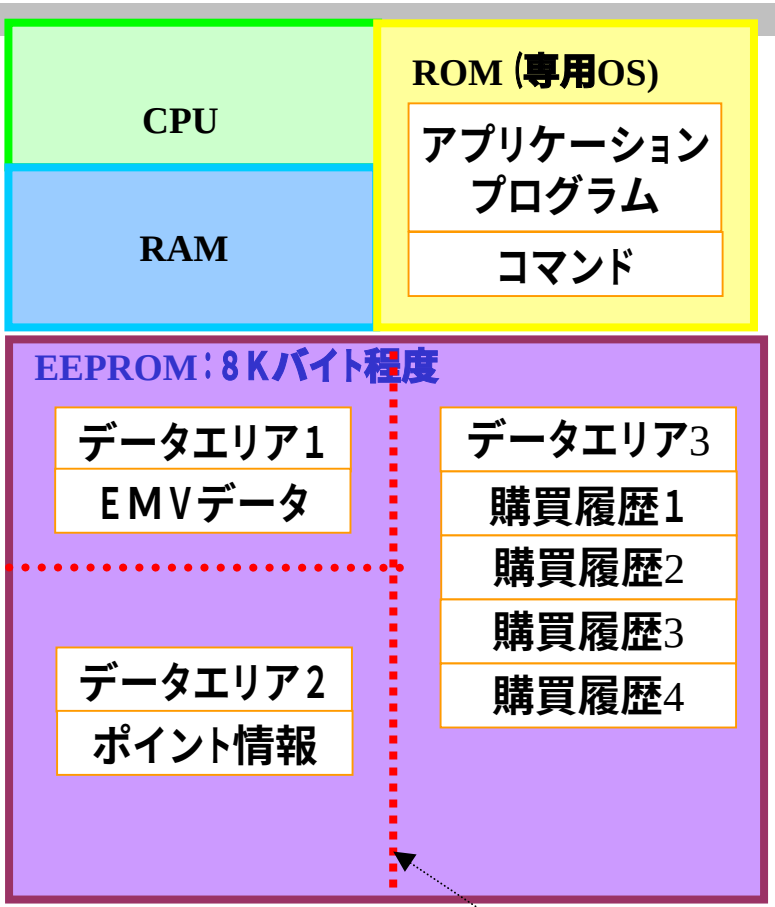
ICカードの種類

- **接触型ICカード**
  - **専用OS (Native OS) : OSとアプリケーションを一体化してROMに格納**
  - **マルチアプリケーションOS : Java Card, MULTOS**
- **非接触型ICカード**
  - **密着型 (通信距離 ～2mm)**
  - **近接型 (通信距離 ～10cm) : Type A, Type B, FeliCa**
  - **近傍型 (通信距離 ～70cm)**
- **ハイブリッドカード**
  - **接触型ICチップと非接触型ICチップを1枚のカードに搭載**
- **デュアルインタフェースカード**
  - **1つのICチップ(メモリ)を2つのI/F (接触I/F、非接触I/F) で共有**

マルチアプリケーションOSと専用OS

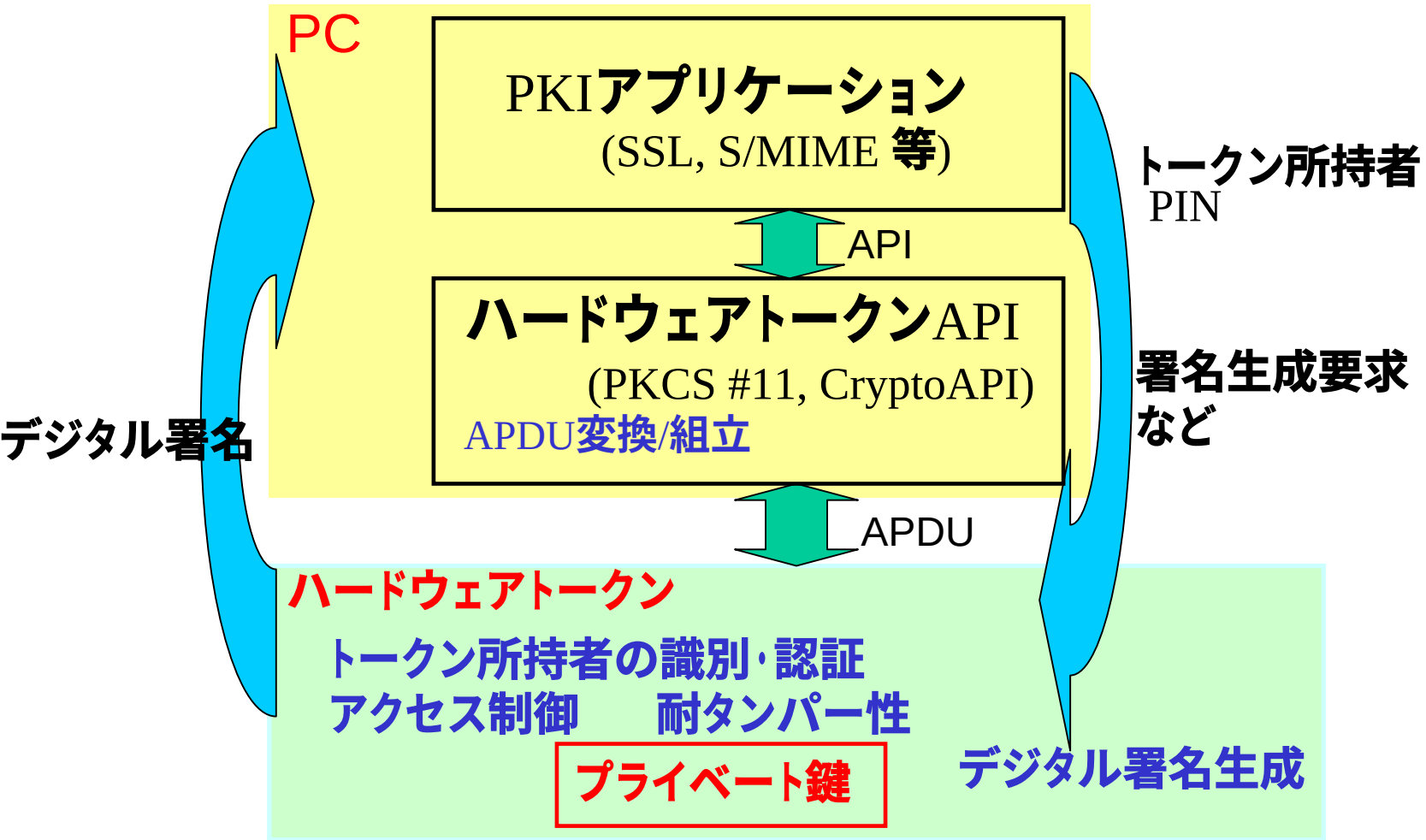


・処理プログラム・コマンド ：アプリケーション毎
：書き換え可能
・データ格納エリア ：アプリケーション毎
：書き換え可能



処理プログラム・コマンド ：チップ共通
：書き換え不可
データ格納エリア ：アプリケーション毎
：書き換え可能

PKIアプリケーションとICカード



なぜハードウェアトークンAPIが必要か?

- ICカードのセキュリティ機能
  - トークン所持者の識別と認証(PINや生体認証)
  - 接続端末の認証(Challenge & Response)
  - アクセス制御
  - 暗号機能(公開鍵ペア生成、デジタル署名生成、等)
  - 暗号化通信(メッセージ認証、暗号化)
- ICカードのセキュリティ機能を利用するためには
  - ICカードとの論理I/F仕様(コマンドAPDU、レスポンスAPDU)
  - ICカードのファイル構造、アクセス権のセキュリティ設定

  などを熟知していなければならない。

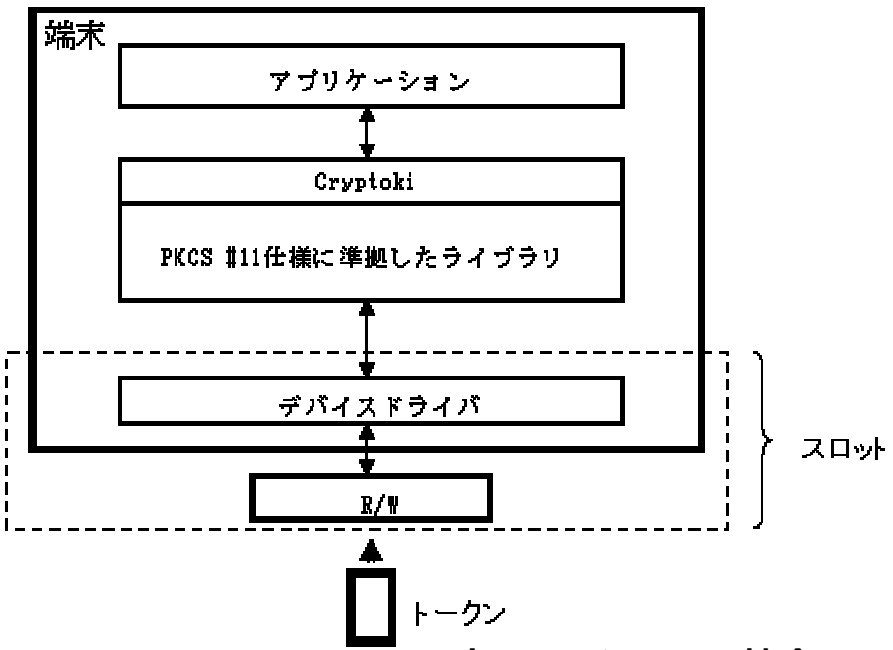
  - 標準化されたハードウェアトークンAPIの利用

PKCS #11

- Cryptokiと呼ばれる
  - 暗号情報を持ち、暗号演算機能を持つデバイスへのAPI
  - 暗号トークンへのインタフェースを規定する
- PKCS #11策定の背景
  - 暗号関連製品の利用を促進させるため、暗号処理におけるトークンインタフェースの標準化
- PKCS #11の目標
  - 特定の機種に依存しない暗号製品の開発促進
  - 複数アプリケーションから複数トークンへのアクセス可能化
- 最新バージョンは v 2.20, 2004年6月28日
  - 本稿では v2.11を解説

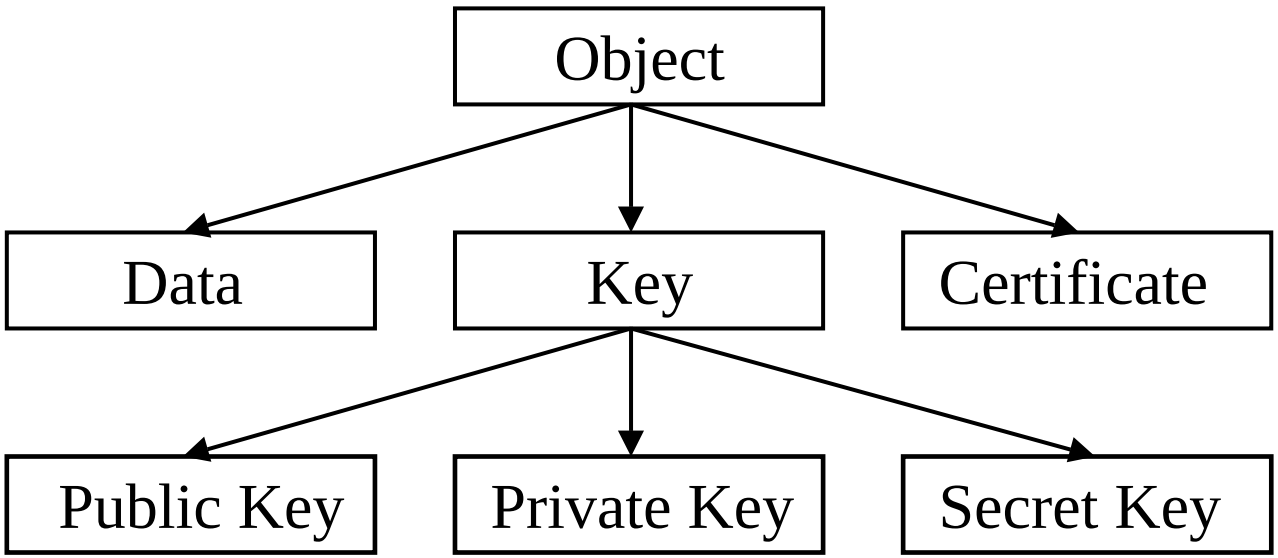
Cryptokiのシステムアーキテクチャ

- アプリケーションがトークンにアクセスする際に「スロット」を検出し、「セッション」と呼ばれる接続を確保して、トークンへのアクセスを行う。
- オブジェクト： トークン内のデータ
- ユーザ：SO(Security Officer)と一般ユーザ



PKCS #11で規定される概念

- **オブジェクト**
  - **トークンに格納されるデータ**



オブジェクトの属性

		属性	説明
一般的な属性(例)		Object Identifier（オブジェクト ID）	トークン内でユニークとするオブジェクトの ID
		Value（値）	オブジェクト内のデータ
種別を表わす属性	セッションにおける種別	Token（永続的）	セッションが終了しても存在する
		Session（一時的）	セッションが終了すると破棄される
	アクセス権における種別	Public（公開）	アクセス権フリー
		Private（秘密）	アクセスに PIN が必要
クラス特有の属性(例)	鍵オブジェクト特有の属性(例)	ID（ID）	鍵の ID
		Start Date（開始日）	鍵の有効開始日
	証明書オブジェクト特有の属性(例)	Issuer（発行者）	発行者の名前

プライベートオブジェクト(Private属性を有するオブジェクト)へのアクセスには、一般ユーザでのPIN認証に成功する必要がある。

セッション

- 暗号処理アプリケーションがトークンにアクセスするための接続を「セッション」と呼ぶ。
- セッションの種類：“読み書き可能”、“読出しのみ”
- 複数のセッションをサポートすることが可能
- セッションの役割
  - ログイン管理
  - オブジェクト管理
  - 暗号処理

PKCS #11 利用上の留意点

- Cryptokiで扱えるスロットは、C_Initialize関数の実行時に確定する。C_Initialize実行後にR/Wを接続しても認識されない。
- 1個のブロックの暗号化では、C_EncryptFinal関数の実行不要
- 戻り値が複数個の関数、例えばC_GetSlotList関数や暗号化関数では、戻り値が複数の場合がある。初めに個数のみを取得し、領域確保後に再度呼び出して値を取得することも可能。

PKCS #11利用手順(例)

- **データオブジェクトをトークン内に作成し、既に格納されている鍵オブジェクトで暗号化を行う場合の例**

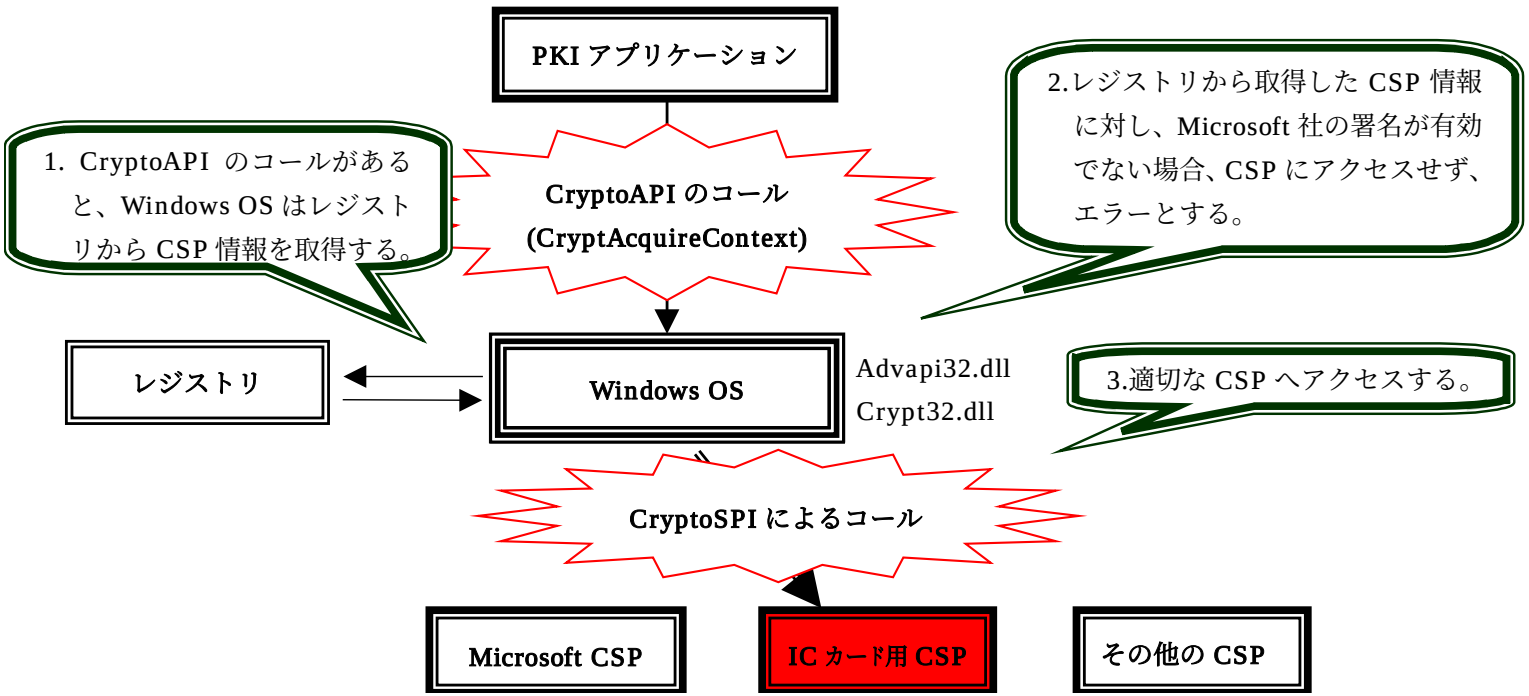
順番	処理内容	使用する関数
1	Cryptoki の初期化	C_Initialize
2	スロットに挿入されているトークンを検出し、それらのスロットを ID として取得する。	C_GetSlotList
3	新しいセッションを開き、セッションハンドルを取得する。	C_OpenSession
4	ログイン処理を行う。その際、取得したセッションハンドルと、PIN を引数として指定する。	C_Login
5	データオブジェクトの作成。その際、セッションハンドルと、オブジェクトの属性を引数として指定する。	C_CreateObject
6	トークン内の鍵オブジェクトを検出するに当たり、そのオブジェクトの属性（例えば、Object Identifier）を指定する。その際、セッションハンドルも引数として指定する。	C_FindObjectsInit
7	指定された属性のオブジェクトを検出する。その際、セッションハンドルと、検出する最大個数を引数として指定すると、オブジェクトのハンドルが取得される。	C_FindObjects

PKCS #11利用手順(例)

順番	処理内容	使用する関数
8	検出の Finalize。その際、セッションハンドルを引数として指定する。	C_FindObjectsFinal
9	暗号化の初期化。その際、セッションハンドルと、検出した鍵オブジェクトのハンドルと、使用する暗号メカニズムを引数として指定する。	C_EncryptInit
10	作成したオブジェクトの Value 属性（データ）に対し、暗号化を行う。その際、セッションハンドルも引数として指定する。	C_Encrypt
11	ログアウト。その際、セッションハンドルを引数として指定する。	C_Logout
12	セッションハンドルを引数として指定し、セッションを閉じる。	C_CloseSession
13	Cryptoki の Finalize	C_Finalize

CryptoAPIとICカード用CSP

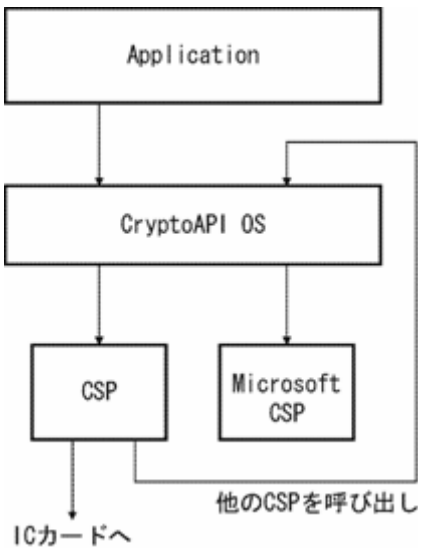
- **暗号アルゴリズムを実装し、暗号鍵を保持するICカード(スマートカード)用のCSP(Cryptographic Service Provider)**



CSPの実装

- **CSPを作成することはCryptoSPIを実装することに他ならない。**
- The Smart Card Cryptographic Service Provider Cookbook

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnscard/html/smartcardcspcook.asp
- **ICカードでサポートしていない機能は、他のCSPを呼び出すことも可能。**



CSP実装上の留意点

- **マルチアプリケーション環境での利用を考慮**
  - **複数アプリケーション**(eg. IEとOutlook)**から同時にICカードが利用される状況を想定してCSPを開発する。**
  - **排他モードで接続する／しない**
- PIN**のキャッシュ**
  - **複数アプリケーションからICカードにアクセスされると、ICカードの認証状態が変更される場合がある。**
  - **ICカードのセキュリティ設定によっては、利用者が何度もPINを入力しなければならない場合がある。**
- **証明書のキャッシュ**
  - **ICカードの通信速度は**9600bps**程度➡ 証明書の読出しに**1～2**秒**

CSP実装上の留意点

- **証明書の登録と利用**
  - Internet ExplorerやOutlook Expressでデジタル証明書が必要な処理(SSLやS/MIME)を行うとき、証明書はローカルストア(レジストリ)に登録されていなければならない。
  - Internet ExplorerやOutlook Expressによる証明書メンテナンス機能は、ローカルストア(レジストリ)に登録された証明書が対象。ICカード用CSPだけでは、ICカード内証明書のメンテナンスができないので注意。
  - ICカード内のデジタル証明書とローカルストア(レジストリ)の証明書の同期を取る仕組みが必要。

CSP, ICカードのレジストリ登録

- CSPやICカード、R/Wに関する情報はレジストリに登録しておく

  a) HKLM¥SOFTWARE¥Microsoft¥Cryptography¥Calais¥SmartCards
  カードのATR、ATRマスク、対応するCSP名を登録する

  b) HKLM¥SOFTWARE¥Microsoft¥Cryptography¥Defaults¥Provider
  CSP名、CSPのプロバイダータイプ、CSPの格納場所（パス）、署名を登録

- レジストリ情報を利用して、ICカード挿入時に自動的に対応したCSPが選択され呼び出される手順の一例

  1. カードが R/W に挿入されると ATR がカードから出力される。
  2. この ATR 情報をキーにして上記 a のレジストリ情報を検索し、挿入されたカードに対する CSP 名を決定する。
  3. この CSP 名を指定して CryptoAPI を呼び出せば、挿入されたカードにマッチした CSP が起動される。
  4. この際、上記 b のレジストリ情報から CSP の実体とその署名データが得られ CSP の正当性もシステムによって検証される。

PKCS #11と CSP の相互運用

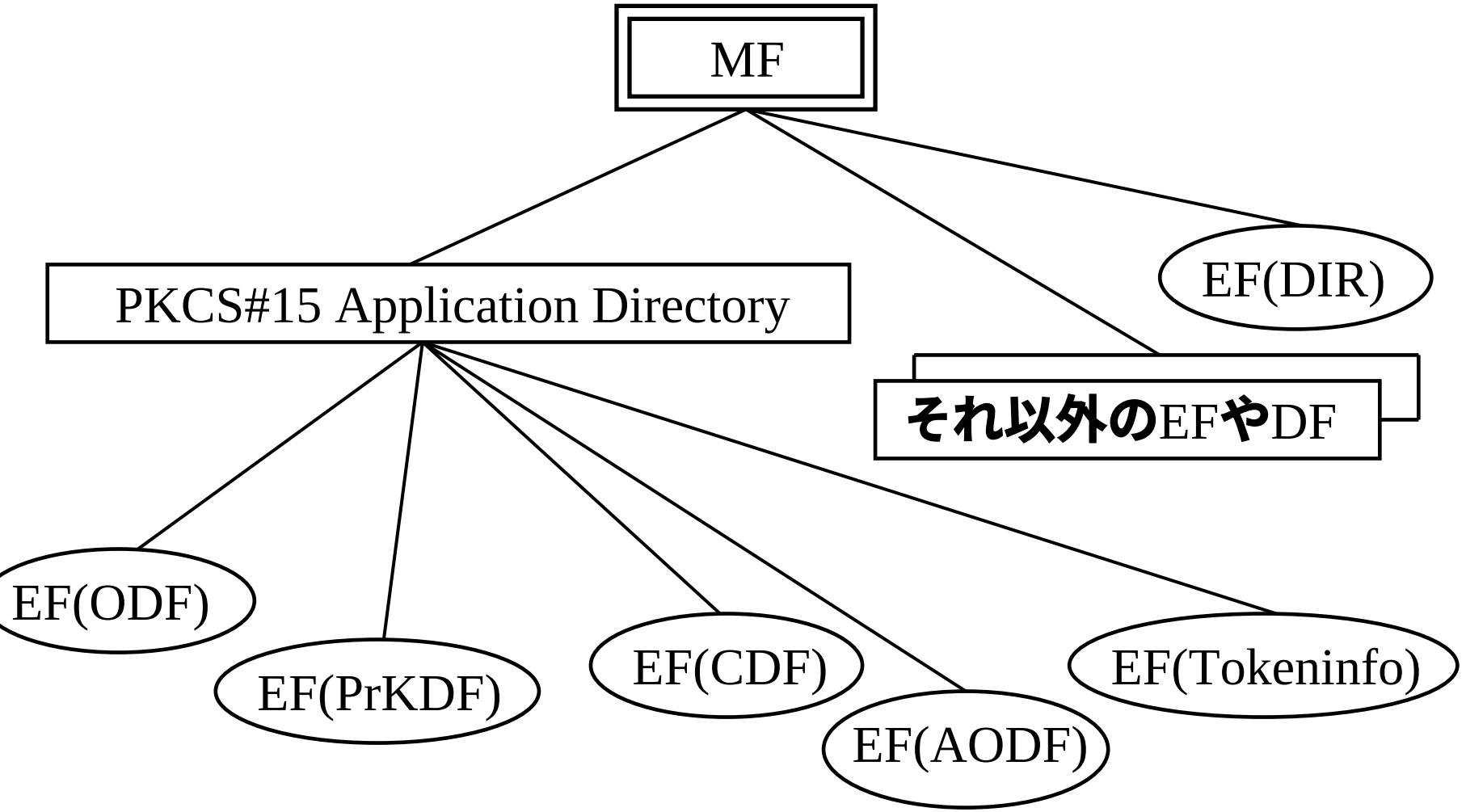
- ICカード内のファイル構造共通化
  - CSPとPKCS #11双方のアクセス性だけを考慮すると、占有するメモリ領域が増大する。
- キャッシュされたPINの共有化
  - マルチアプリケーション環境では、他のアプリケーションがICカードの認証状態を変化させることがある。利用者の利便性のためにPINをキャッシュする実装では、キャッシュされたPINへのアクセス制御の仕組みを考慮する必要がある。
- データの差異の吸収
  - CSP固有の属性：「コンテナ名」と「鍵のスペック」
- 属性証明書
  - CSPには属性証明書を扱うための仕組みは提供されていない。

PKCS #15

- **暗号トークン(ICカード等)内のファイル構造に関する標準化**
  - **トークン上のセキュリティ情報格納先のファイル／ディレクトリの標準フォーマット**
  - ISO/IEC 7816-15 : Information technology - Identification cards - Integrity circuit(s) cards with contacts - Part 15: Cryptographic information application
- **オブジェクト･モデル**
  - **鍵オブジェクト**
    - private key, public key, secret key
  - **証明書オブジェクト**
    - X.509 Certificate
  - **認証用オブジェクト**
    - PIN Object, Biometric Template
  - **データオブジェクト**

サブクラスの実体が、ICカード内のファイルに相当する。

PKCS #15 / ファイル構造のモデル



PKCS #15 / ファイル構造のモデル

- DF : Directory File
  - カード内オブジェクトと実際のファイルフォーマットをつなぐレイヤ。ASN.1で記述する。
- EF(ODF) : Object Directory File
  - PKCS#15 Application Directory下にあるオブジェクトへの参照ポインタ
- EF(PrKDF) : Private Key Directory File
- EF(CDF) : Certification Directory File
- EF(AODF) : Authentication Object Directory File
- EF(TokenInfo) : トークンに関する情報(シリアルNo.等)を保持

PKCS #15 の機能 / オブジェクトの管理

- **オブジェクトの追加･削除時は、**Directory File**もメンテナンス**
  - **追加したオブジェクトに該当する**Directory File**にポインタを追加する。**
  - **オブジェクト削除時、削除したオブジェクトに該当する**Directory File**の削除位置を指すポインタを**‘00’**で更新し、**EF(UnusedSpace)**にポインタを追加する。**

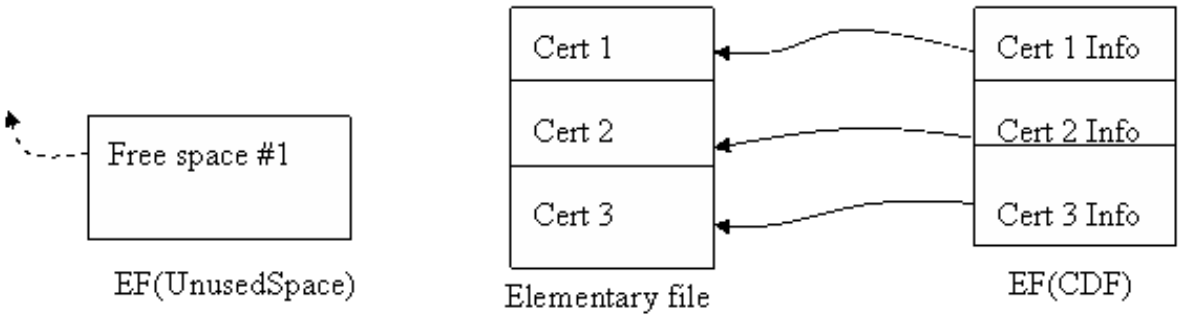


図 6-5 オブジェクト削除の例：削除前

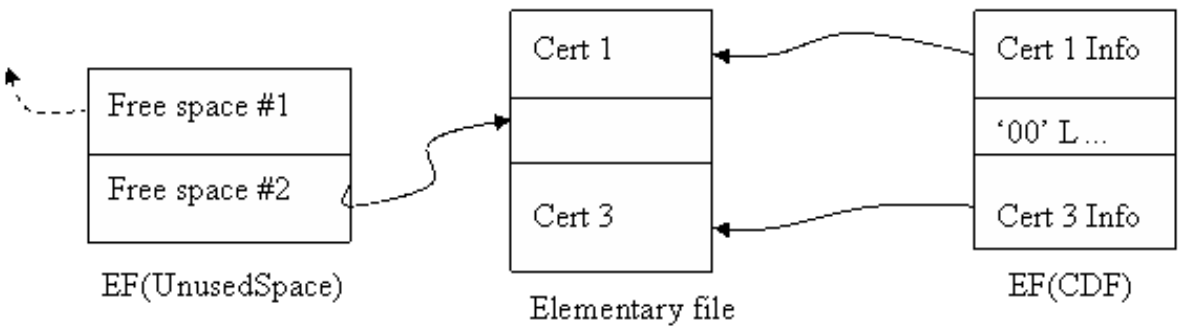
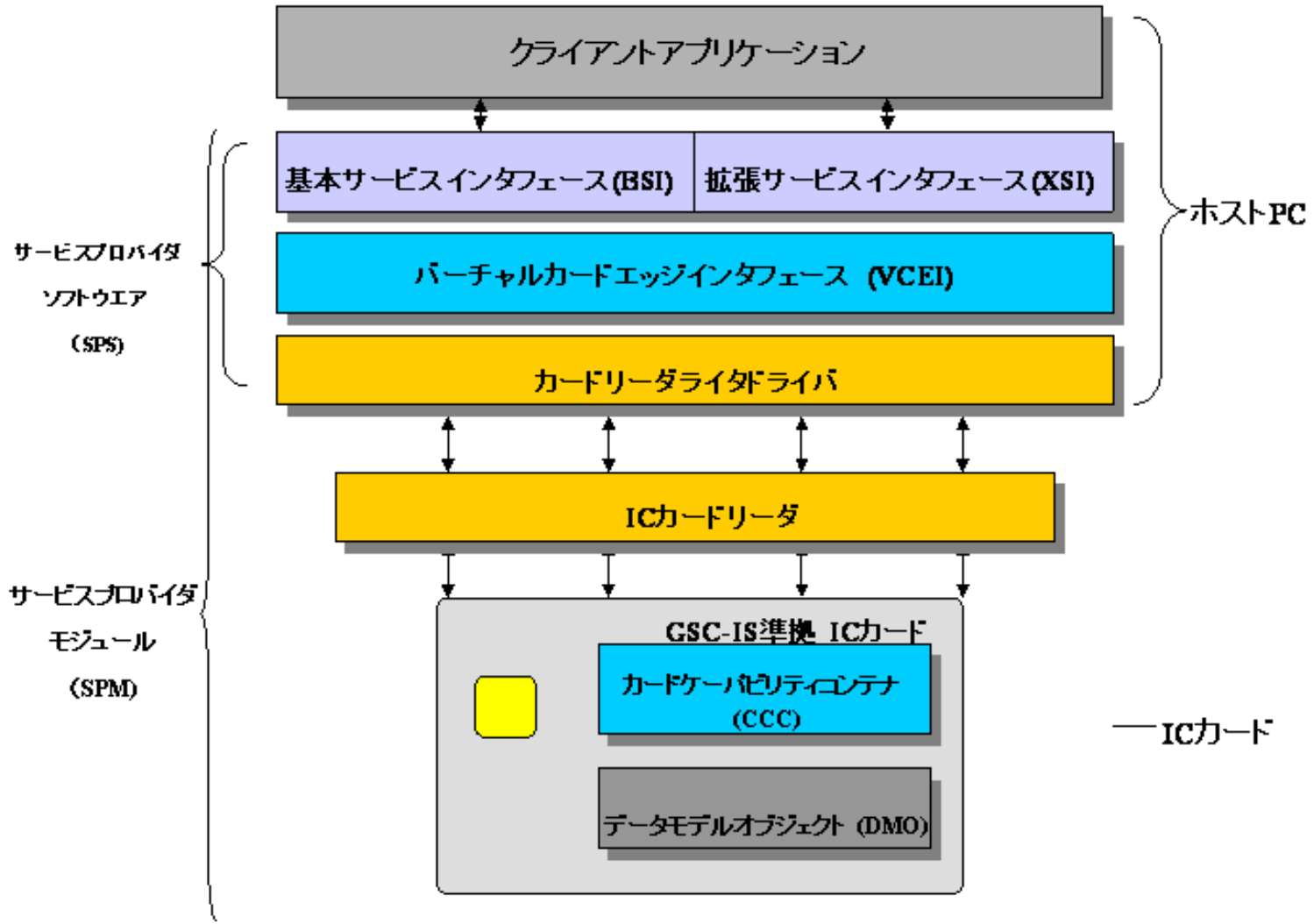


図 6-6 オブジェクト削除の例：削除後

GSC-IS

- **背景**
  - IC**カードを用いたシステムの相互互換ソリューションとして、**GSA (General Services Administration)**と** NIST **によって**Government SmartCard Interoperability Specification (GSC-IS)**が策定された。**
- GSC-IS**の構成**
  - ServiceCallLevel
    - BSI(Basic Services Interfaces)**と呼ばれる**API**を定義**
    - SPM(Service Provider Module) :**サービスを提供する下位レイヤ** = IC**カード** + R/W + **通信**S/W (SPS)
  - CardCommandLevel
    - **カードへ送る**APDU**を定義。**
    - VCEI (Virtual Card Edge Interface)

GSC-ISアーキテクチャモデル



まとめ

- ICカードのセキュリティ機能をPKIアプリケーションから利用するため、ハードウェアトークンAPIの標準化が進んでいる。
- ハードウェアトークンAPI
  - PKCS #11
    - Netscape Navigator, Netscape Messanger, Entrust Products
  - CryptoAPI (ICカード用CSP)
    - Microsoft Internet Explorer, Microsoft Outlook Express, Outlook
  - GSC-IS
- 暗号トークン内のファイル構造
  - PKCS #15