



INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN

# IC・ID カードの相互運用可能性の向上に係る基礎調査

シーズ編

報告書

2007 年 1 月

独立行政法人 情報処理推進機構

- \* 「FeliCa」は、ソニー株式会社の登録商標です。
- \* 「Solaris」、「Java Card」、「Java」およびすべての Java 関連の商標は、米国およびその他の国における米国 Sun Microsystems, Inc.の商標または登録商標です。
- \* 「MULTOS」は、MAOSCO の登録商標です。
- \* 「Windows」、「MS Windows」は、Microsoft Corporation の米国およびその他の国における登録商標および商標です。
- \* 「Linux」は、Linus Torvalds 氏の米国およびその他の国における登録商標および商標です。
- \* 「Mac OS X」、「Mac」、「Safari」は、Apple Inc.の米国およびその他の国における登録商標及び商標です。
- \* 「Cryptoflex」、「Cyberflex」、「Reflex USB」は、Axalto 社の登録商標です。
- \* 「Micardo」は、Sagem Orga 社の登録商標です。
- \* 「eToken」は、Aladdin Knowledge Systems 社の登録商標です。

## 目次

<b>1. IC・IDカードの相互運用可能性の概要</b> .....	<b>1</b>
1.1. はじめに .....	1
1.2. 基本的な用語 .....	1
1.3. 報告書の意図と構成 .....	2
1.4. IC・IDカードサービスに要求される機能 .....	6
1.4.1. IC・IDカードを用いた認証(Authentication) .....	6
1.4.2. IC・IDカードに格納されるデータ .....	8
1.4.3. IC・IDカードに格納されるデータに対する操作 .....	12
1.5. IC・IDカードの相互運用可能性の課題 .....	14
1.5.1. IC・IDカードの相互運用可能性の分類 .....	14
1.5.2. 相互運用可能性の課題 .....	17
1.5.3. データモデルの課題 .....	19
1.5.4. カードエッジ・インターフェースの課題 .....	22
1.5.5. ミドルウェアの重要性 .....	23
1.6. IC・IDカードの実装、展開 .....	25
<b>2. 標準化動向に関する調査</b> .....	<b>29</b>
2.1. ICカードサービスの概要 .....	29
2.1.1. ISO/IEC 24727-1 .....	31
2.1.2. ISO/IEC 7816 .....	34
2.2. ICカードOS .....	35
2.3. セキュリティ保持の仕組み .....	36
2.3.1. アプリケーション識別子(AID) .....	37
2.3.2. セキュリティ属性 .....	39
2.3.3. セキュリティ環境 .....	40
2.4. カードエッジ・インターフェース.....	42
2.4.1. SELECT コマンド .....	45
2.4.2. データ操作コマンド .....	46
2.4.3. 認証用コマンド .....	46
2.4.4. VERIFY コマンド .....	48
2.4.5. MANAGE SECURITY ENVIRONMENT コマンド .....	48
2.4.6. PERFORM SECURITY OPERATION コマンド .....	49

2.4.7.	ISO/IEC 24727-2 で利用されるカードコマンド群 .....	50
2.5.	データモデル .....	52
2.5.1.	ISO/IEC 7816-15 .....	53
2.5.2.	ISO/IEC 7816-15 と PKCS#15 の差異 .....	59
2.6.	カードサービスの API .....	60
2.6.1.	ISO/IEC 24727-3 .....	61
2.6.2.	PKCS#11 .....	61
2.6.3.	24727-3 と PKCS#11 の違い .....	62
<b>3.</b>	<b>IC・IDカードの実装例 .....</b>	<b>63</b>
3.1.	IDカードの動向 .....	63
3.2.	FINEID .....	63
3.2.1.	プロジェクト概要 .....	63
3.2.2.	技術概要 .....	63
3.2.3.	カードコマンドとデータ構造、アクセス管理 .....	65
3.3.	BELPIC .....	71
3.3.1.	プロジェクト概要 .....	71
3.3.2.	技術概要 .....	72
3.3.3.	カードコマンドとデータ構造、アクセス管理 .....	73
3.4.	PIV .....	77
3.4.1.	プロジェクト概要 .....	77
3.4.2.	技術概要 .....	78
3.4.3.	カードコマンドとデータ構造、アクセス管理 .....	79
3.4.4.	PIV カードアプレットの参考実装 .....	83
3.4.5.	PIV の今後の拡張 .....	86
3.5.	カードコマンド、データモデルの比較 .....	87
3.5.1.	カードコマンドの比較 .....	87
3.5.2.	データモデルの比較 .....	89
<b>4.</b>	<b>ミドルウェアの実装例 .....</b>	<b>91</b>
4.1.	ミドルウェアの実装 (OPENSC) .....	91
4.1.1.	OpenSC プロジェクトの概要 .....	91
4.1.2.	全体のアーキテクチャ .....	94
4.1.3.	カードエッジインタフェース .....	96
4.1.4.	データモデルとの対応付け .....	103

4.1.5.	プライベート鍵の操作と API	113
4.1.6.	各種 IC・ID カードの対応	114
4.1.7.	署名操作の流れ	123
4.1.8.	PKCS#15 データモデルの実装とパフォーマンス	127
4.1.9.	マルチプラットフォーム対応	129
4.2.	PIV	130
4.2.1.	全体のアーキテクチャ	130
4.2.2.	クライアント API 実装	131
<b>5.</b>	<b>テストと認定制度</b>	<b>136</b>
5.1.	米国の PIV の成り立ち	136
5.1.1.	HSPD-12	136
5.1.2.	FIPS-201	138
5.1.3.	HSPD-12 実施のロードマップ	141
5.2.	PIV のテスト方法論と仕様	142
5.2.1.	テスト方法論	143
5.2.2.	SP800-85A	144
5.2.3.	SP800-85B	146
5.2.4.	PIV のテストツール	148
5.3.	PIV の認定制度と製品	150
5.3.1.	PIV の認定制度	150
5.3.2.	PIV の認定製品	152
	<b>参考文献リスト</b>	<b>155</b>

# 1. IC・ID カードの相互運用可能性の概要

## 1.1. はじめに

これまで IC カードと言えば、「モノとしての IC カード」の側面に焦点が当てられることが多かった。普及させるための様々な取り組みも「モノとしての IC カード」中心に物理的な形態の規格等に行なわれてきた。

本報告書では、単体の「モノとしての IC カード」よりも、IC カードに格納されたデータフォーマットもしくはデータモデルと、このデータモデルを扱う端末側のミドルウェアに焦点を当てている。IC カードを使った ID カード (IC・ID カード) を IT 社会における基盤に組み入れようとする動きが世界中至るところで見られる。このとき IC・ID カードは、フロントエンドツールであり、このフロントエンドツールはバックエンドのシステムとうまく融合する必要がある。そのためには、IC・ID カードを扱う環境が整備される必要があり、その際の大きな課題に IC・ID カードとミドルウェアから構成された IC・ID カードサービスとしての相互運用可能性の確保がある。「モノとしての IC カード」に対してソフトウェア・アーキテクチャが重要な要素となるミドルウェアの実体を理解することは難しい面がある。しかし、ミドルウェアは、フロントエンドツールとしての IC・ID カードと、バックエンドシステムの重要な糊付け役を担う。ミドルウェアも含んだ相互運用可能性の問題に関して、これまであまり認識されていない面があるが、この問題の解決なくしては、IC・ID カードが IT 社会における基盤として利用され普及することは考えにくい。本調査報告書 (シーズ編) では、こうした問題の解決するための技術的な課題を整理し、相互運用可能性に関連した、標準化動向、実装事例などを示すことにより今後の方向性を示唆する。

## 1.2. 基本的な用語

表 1 基本的な用語

用語	意味
IC・ID カード	IC カードを使った ID カード
カードアプリケーション	IC カード上のアプリレット、データ
ミドルウェア	IC・ID カードを扱うための端末上のミドルウェア
IC・ID カードサービス	IC・ID カードとミドルウェア

カードエッジ・インターフェース	IC・ID カードとミドルウェア間の論理的なコマンドインターフェース APDU(アプリケーションプロトコルのデータ・ユニット)を使ってやり取りする。
データモデル	IC カード上のファイル構造、データ構造

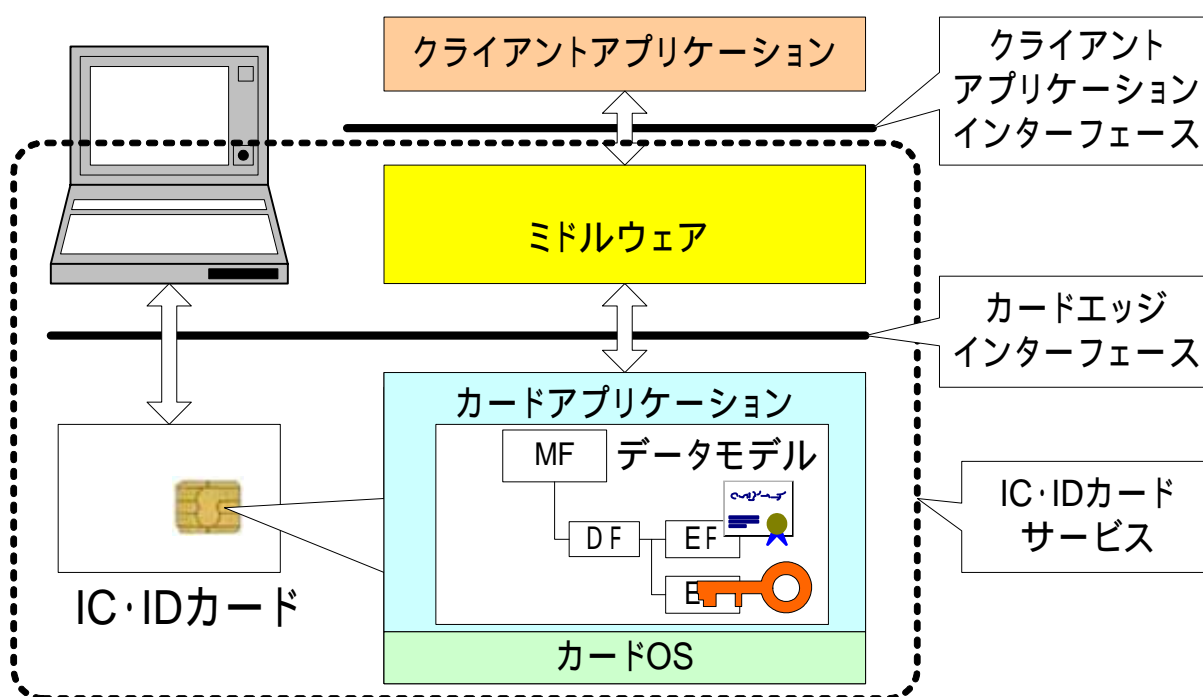


図 1 基本的な用語

### 1.3. 報告書の意図と構成

「IC・ID カードの相互運用可能性向上に係る基礎調査報告書・シリーズ編」は、以下のようなことを意図して書かれている。

(1) IC・ID カード(サービス)の全体像を簡潔に記述し、相互運用可能性の問題を分かり易く説明

PKI を扱う IC・ID カード、IC・ID カードに対応したミドルウェア、PKI アプリケーションの関係を簡潔に説明し、その上で相互運用可能性の問題を分かり易く説明する。

相互運用可能性の問題は、技術全体を把握することが難しいこともあり、ベンダー等の供給者側が解決する技術の問題だと見なされているところがある。しかし、技術を比較よく理解している供給者側は、必ずしも相互運用可能性の問題を積極的に解決するモチベーションがある訳ではない。実際、相互運用可能性の問題の解決でベネフィットがあるのは利用者側であることが多い。また、供給者側は、普及のために相互運用可能性の問題の解決を願っているかもしれない。しかし供給者側の積極的な活動は、顧客の囲い込み戦略になる場合もあり、利用者側からは、むしろ相互運用可能性の問題の解決を阻害していると思なされる可能性も否定できない。供給者側にとっても、IC・ID カードを広く普及させるために相互運用可能性の問題の解決が重要であるが、そのためには、利用側のイニシアチブが非常に重要になる。

利用者、調達側からすると、そもそも「何が相互運用可能性の問題か」ということ自体を理解することが非常に難しい。一般的に利用者は、世の中にあるものを受け入れるしかない。標準化されて相互運用可能性の問題が解決できるとしても、高い対価を払ってまで、そうした製品を購入するには至らないかもしれない。

このような問題の解決のためには、調達側、利用者側のトップダウンな方針が重要だが、マルチセクター、マルチベンダー、マルチプラットフォームと、利用範囲、応用範囲が広がる程に、技術全体を理解することも、関係者（ステークホルダー）を取りまとめることも難しい。しかし、利用範囲、応用範囲の広がりこそが IC・ID カードのあるべき方向性であり目標のはずである。IC・ID カードにおいても、相互運用可能性の確保の解決には、まずは、共通認識としての全体のアーキテクチャと相互運用可能性の課題をステークホルダー間で共有する必要がある。

## (2) ミドルウェアの重要性の説明

これまで IC カードに関しては、「モノとしての IC カード」自体と、そのカード上のアプリケーションに焦点が当てられることが多かった。また、ミドルウェアは、IC カードの標準化の範疇ではなかったこともあり相互運用可能性の課題も認識されていなかった。しかし、PKI（公開鍵技術基盤）を扱う IC・ID カードの場合、端末側のミドルウェアを含めた IC・ID カードサービスとしての相互運用可能性に大きな課題がある。また、このミドルウェアの持つ機能や、ミドルウェアのマルチプラットフォーム対応などが重要な意味をもつことが多い。

「モノとしての IC カード」に対してソフトウェア・アーキテクチャが重要なミドルウェアの実体を理解することは難しい面もある。本報告書では、ミドルウェアのオーブ

ソースの実装例等も示すことによって、その重要性を説明する。

(3) 調達者、認証システムの設計者、開発者等の視点を重視し、標準、仕様、実装を説明

相互運用可能性の課題に対して標準、仕様、実装をバランスよく説明する。これには2つの意味がある。1つめは、相互運用可能性の課題をいくつかの角度から考察すること、2つめは、標準から実際に相互運用可能性を確保するためのプロファイル(ないし最小限の要求仕様)を抽出することである。

2つめに関して、実際、相互運用可能性の課題を標準仕様からだけで説明することは非常に難しい。また、相互運用可能性の解決には、標準仕様の観点からだけでなく、実際に展開されているIC・IDカードの仕様や、実際の実装を的確に把握することが重要になる。標準仕様と言う側面からだけのアプローチでは、机上の空論になりかねない。こうしたことから本報告書では、IC・IDカードの相互運用可能性と関連性の深い標準、仕様が公開されているIC・IDカードの仕様、ソースコードが公開されているミドルウェアの説明を行なうことで、相互運用可能性の課題を説明する。

2つめに関して、相互運用可能性の問題を解決するIC・IDカード、ないし、IC・IDカードサービスの仕様は、標準仕様を中心に策定されるべきである。しかし多くの標準は汎用的な目的で作成されているため、多くのオプションをもつ。現実的に多くの標準は、そのままでは現実的な開発量で実装ができるものではなく、標準に従うというだけでは、相互運用可能性の問題の解決にはならない。実際のIC・IDカードの仕様例や実装例を調査することにより、標準仕様の中のどの規定をどこまで実装するのがよいのかといったことを考察する。

(4) 今後の方向性の示唆

(高機能な)IC・IDカードが広く利用されるようにするためには、「カードエッジ・インターフェース」「データモデル」「クライアントアプリケーション・インターフェース」の標準化が促進され、IC・IDカード、ミドルウェアの各コンポーネントが独立性をもつ方向へ開発されるべきであると考えられる。このようなことを実現するためには、幅広く利用されるためのIC・IDカードの仕様の策定と公開、更にこれらの仕様に基づいたテスト仕様の作成、レファレンス実装やテストツールの開発なども検討されるべきと考えられる。事例等を説明することにより今後の方向性を示唆する。

このような本書の趣旨に照らして、1章(本章)では、IC・IDカード(サービス)の全体像を簡潔に記述した上で相互運用可能性の課題を説明する。表2、図2に報告書全体の構成を示す。

表2 報告書の構成

章	大項目	中項目	コンセプト&内容
1	概要	ICカードの概要 相互運用可能性の概要	IC・IDカードサービスに要求される機能、アーキテクチャ等をバランスよく記述し、その上で、相互運用可能性の課題の概要を説明する。
2	相互運用可能性に関連した標準	標準化動向 ISO/IEC 7816-4,8,9 ISO/IEC 7816-15、 ISO/IEC 24727	相互運用可能性に関連したIC・IDカードサービスの標準を簡潔に説明する。特に3章、4章を説明する上で理解が欠かせない部分について説明する。
3	IC・IDカードの実装例	IC・IDカードの動向 FINEID(フィンランド) BELPIC(ベルギー) PIV(米国)	事例研究として、相互運用可能性を配慮して発行されているIC・IDカードの実際の仕様を説明する。
4	ミドルウェアの実装例	OpenSC(オープンソース) PIV(米国)	事例研究として、ミドルウェア実装を説明する。実際の実装が相互運用可能性の問題にどの様に対処しているかに焦点を当てる。
5	テストと認定制度	PIV(米国)	事例研究として、PIV/FIPS201のテスト仕様、テストツール、認定、「認定製品の例」を説明する。

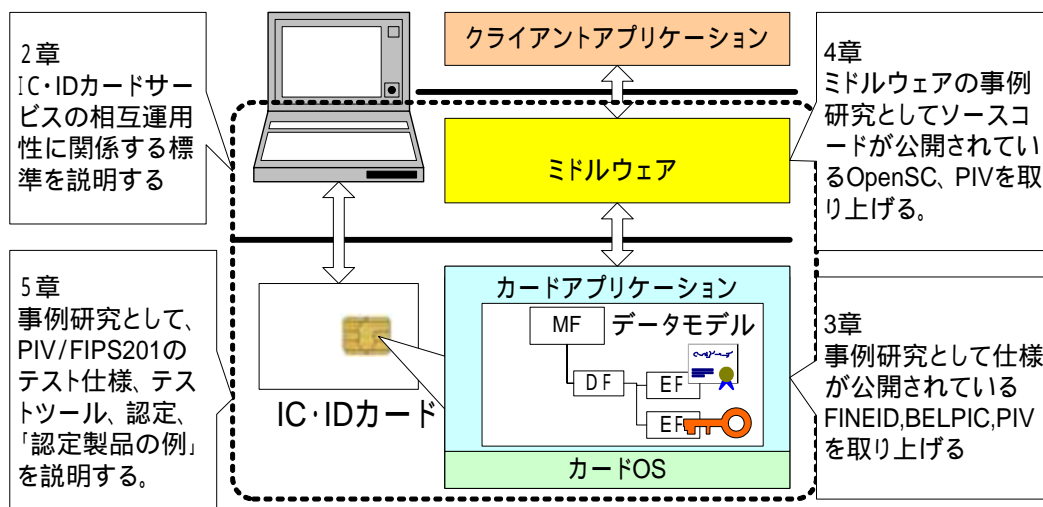


図 2 報告書の構成

## 1.4. IC・ID カードサービスに要求される機能

### 1.4.1. IC・ID カードを用いた認証(Authentication)

一般的な IC・ID カードを実現する基本的な技術に PKI（公開鍵技術基盤）がある。公開鍵暗号では、プライベート鍵と公開鍵の 2 つの鍵が使われる。文書等への署名にプライベート鍵が使われ、署名の検証をもう一方の公開鍵を使って行うことができる。IC・ID カードを利用したリモート認証では、IC・ID カードに格納された鍵（プライベート鍵）の署名（演算）が利用される。図 3 に IC・ID カードを利用した典型的なチャレンジ・レスポンスによるリモート認証の例を示す。

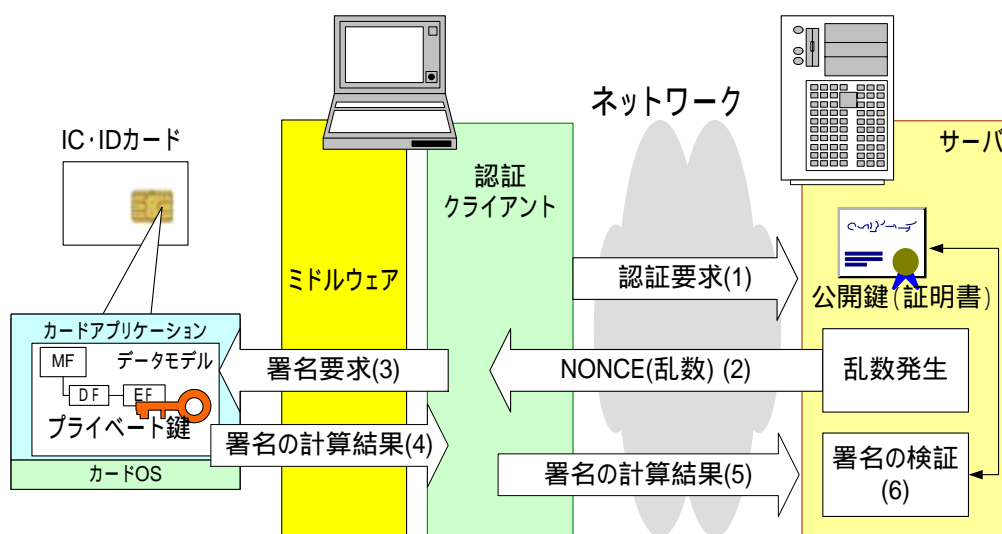


図 3 リモート認証

以下にリモート認証の手順を示す。

- (1) 認証 ( Authentication ) を要求する認証クライアントは、サーバに対して認証要求を行なう。
- (2) サーバは認証要求に対して、この場限りの値である NONCE を生成し認証クライアントに送る。
- (3) 認証クライアントは、サーバから受け取った NONCE を元に IC・ID カードに対して署名要求を行なう。
- (4) IC・ID カードは、この NONCE に対してカード上で署名を施し、署名結果を返す。
- (5) 認証クライアントは、受け取った署名結果をサーバに返す。
- (6) サーバは、認証クライアントから送られてきた NONCE に対する署名を、公開鍵を使って検証する。

この例に示すように、 IC・ID カードの秘密情報 ( プライベート鍵 ) は IC・ID カードから出ることは無く、ネットワーク上に流れることも無い。また、 IC・ID カード保有者の秘密情報が、サービス側のサーバに格納されないことも重要な点である。サービス提供者は、 IC・ID カード保有者の秘密情報 ( 例えばパスワード ) を預かる必要がない。これは IC・ID カード保有者にとっても、サービス提供者にとっても大きなメリットとなる。

このとき、署名されるデータ ( この場合 NONCE ) 自体は認証プロトコル上の重要な意味を持つが、 IC・ID カード保有者にとって意味があるデータではない。この署名は、

カード保有者の意志を表明する訳ではない点に注意する必要がある。

以上のように IC・ID カードを用いたリモート認証においてカード内のプライベート鍵による署名が使われるが、このプライベート鍵の保護は、セキュリティ上重要な要件になる。

IC・ID カードとして用いられる IC カードは、署名操作などで利用される暗号演算機能を有する他、アクセス制御ルールを設定したり、(カード内のメモリ上データの不正な読み出しや改ざんを困難とするだけでなく、物理的な攻撃も含む様々な攻撃から内部データを守る)「耐タンパー性」をもつ。この「耐タンパー性」機能によって、確実なカード保有者認証を実現することになる。

IC・ID カード自体は、多くの場合、PIN による知識認証を行なうことにより IC・ID カード保有者の認証を行なう。IC・ID カードは、「カード保有者の認証を行った上で、プライベート鍵による署名などの暗号演算を許可する」等のアクセス制御ルールが設定される。この様に IC・ID カードは、「耐タンパー性」と適切なアクセス制御ルールの設定により、プライベート鍵などのセンシティブ情報を守ることになる。

#### 1.4.2. IC・ID カードに格納されるデータ

耐タンパー性をもった IC・ID カードにプライベート鍵が格納され、そのプライベート鍵の演算によりリモート認証を安全に行なえることを説明したが、IC・ID カードは、認証(Authentication)だけでなく、電子署名にも用いられる。IC・ID カードを利用した文書への署名(ここでは自署名と表現する)と認証 ( Authentication ) は、共にプライベート鍵による署名(プリミティブな操作としての署名を単に「署名」と表現する)を利用して実現される。しかし、自署名と認証では、そのプライベート鍵による署名の意味が大きく異なる。

図 4 に署名と(リモート)認証の鍵を使い分けている例を示す。

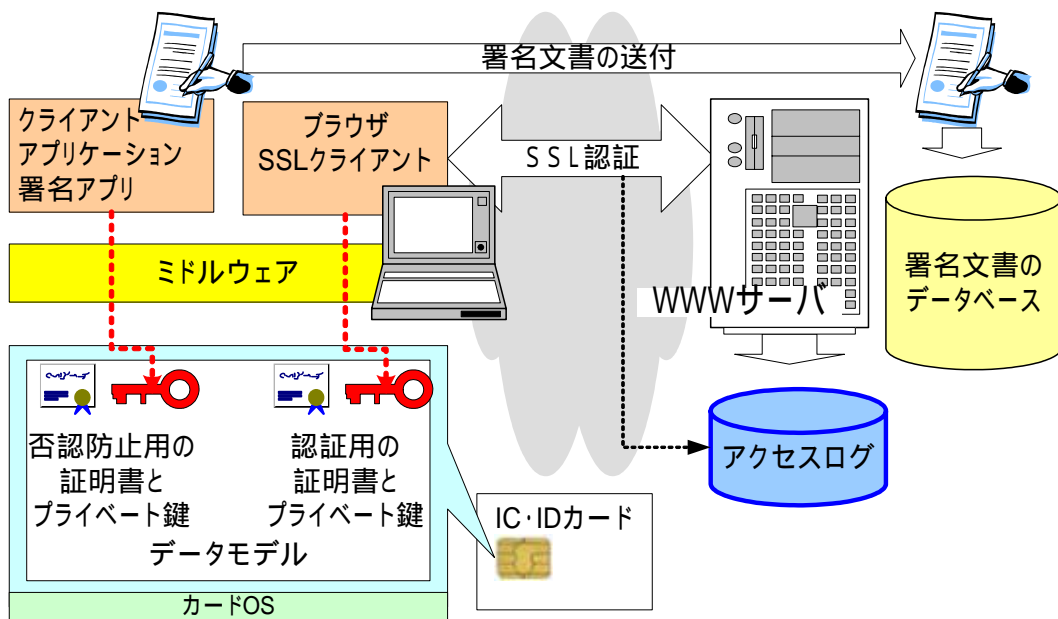


図 4 リモート認証と否認防止の署名

ここで IC・ID カード保有者は、カードに格納された 2 つのプライベート鍵による署名を使い分け、「クライアント認証」と「文書への電子署名」を行っている。IC・ID カード内のプライベート鍵による署名操作は、強い認証(Strong Authentication)機能を実現する。そして、この強い認証を利用することによってサーバへ電子文書をセキュアに渡すことができ、サーバ側では認証のアクセスログを残すことができる。しかし、それだけでは、電子契約などで要求される「実印での捺印」の代わりにはならない。契約文書などに自署名を施す際、IC・ID カード保有者は、この文章の内容を熟読した上で自分の意志を持って自署名を行う。

認証(Authentication)のみのシステムでは、電子文書というトランザクションと利用者の紐付けの証跡をログという形で残すが、これは、サービス提供者と利用者が利害関係にある場合、否認防止にはならない。サービス提供者が利用者の自署名を付した電子文書受け取った場合、この自署名付き電子文書を保存することで、サービス提供者は、利用者の否認防止を行うことができる。このように、否認防止性が重要な利害関係者間の文書のやり取り等では、IC・ID カード保有者の自署名が施された電子文書自身が相手に送付され、その署名された電子文書自体が保存されることが重要になる。このような自署名は、否認防止の署名と呼ばれる。

PKI では、この署名に使われるプライベート鍵に対応する公開鍵を証明するための電子的な証明書である公開鍵証明書が使われるが、IC・ID カードを用いる際には、IC・ID カード保有者のプライベート鍵と共に、このカード保有者の公開鍵証明書(以後、証

明書)が格納される。この IC・ID カード保有者の証明書には、この証明書に対応したプライベート鍵の使用目的が記述されている。否認防止目的で使用される証明書には、証明書に含まれる証明書拡張フィールドの鍵使用目的(Key Usage)に、否認防止用を示すための non-repudiation(否認防止) bit が設定される。non-repudiation bit が設定された証明書に対応するプライベート鍵で(否認防止のための)署名を行う場合、そのアプリケーションは必ず署名者、すなわちカード保有者に自署名する文書を提示する必要がある。

自署名と認証では、処理手順のみならず、想定される脅威も異なる。ネットワーク社会において、なりすましや盗聴といった脅威が語られるが、自署名に対する脅威にもうひとつ、「内容を理解せず(させずに)自署名を行う(行わせる)」という脅威がある。例えば、「手形の裏書の意味を知らずにいわゆる自署名をさせられた」というようなことが起こり得る。

IC・ID カードの PKI 機能を利用した認証においては、その認証プロセスの中で乱数などに署名させて、その署名結果を検証することで認証を行なう。認証のための署名においては、利用者は署名対象(認証プロトコル中の乱数など)を確認することはなく、また、認証のプログラムも利用者に意識させずに署名操作を行うことが多い。それに対して自署名では、署名者が必ず自署名の対象となる文書を確認する必要がある。

以上のようなことから IC・ID カードにおいては、複数の証明書とプライベート鍵を格納して、自署名や認証などの用途に応じて使い分ける例が多く見受けられる。

3章で取り上げているフィンランドの FINEID、ベルギーの BELPIC といった欧州の市民向けの電子的な身分証明書カードは、eID と呼ばれている。この eID や、同じく 3章で取り上げている米国の政府職員向けに発行される PIV(Personal Identity Verification:個人 ID 認証)等では、複数の証明書とプライベート鍵が IC・ID カードに格納され、このプライベート鍵を保護するためのメカニズム、すなわちアクセス制御ルールの設定も異なっている。

FINEID や BELPIC といった欧州の eID の場合、認証用のプライベート鍵による署名では、IC・ID カード保有者がカードの PIN を入力し保有者認証を行なった以降は、カードに格納された認証用のプライベート鍵が認証要求の都度自動的に署名する。これに対して自署名のプライベート鍵では、1回の自署名操作、つまりひとつの文書の自署名毎に「カード保有者の同意確認」(User Consent)のための PIN の入力が必要な仕様になっている。これはカード自体が、「内容を理解せずに自署名してしまうこと」を防ぐ仕組みを有していると言える。

欧州の eID では、IAS すなわち Identification、Authentication と electronic

Signature と言ったコンセプトで仕様が作成されており、電子署名、電子認証に利用できるものとなっている。

日本の公的個人認証サービスでは、証明書の non-repudiation(否認防止) bit が設定された否認防止目的の証明書のみが発行されている。従って、公的個人認証サービスの発行する証明書を、電子認証(Authentication)としての利用には注意が必要である。

以上の様に IC・ID カードに格納されるデータとしては、IC・ID カード保有者のプライベート鍵と証明書があるが、これは、目的に応じて複数セット格納される場合があり、また、目的に応じたアクセス制御ルールの設定がなされていることに注意すべきである。

表 3 に、FINEID、BELPIC、PIV、公的個人認証サービス(JPKI)の 4 種類の IC・ID カードに関して、プライベート鍵と証明書の扱いについての比較を示す。

表 3 プライベート鍵と証明書の扱いの比較

IC・ID カード	カード保有者の証明書 とプライベート鍵の種類	説明
FINEID フィンランド	認証、暗号用の 証明書	( 認証のための ) 署名操作と暗号文からの復号に使用。 PIN 1 によるカード保有者の認証。
	否認防止の署名用証明書	署名操作のみ。署名操作毎に PIN 2 によるカード保有者の認が必要。
BELPIC ベルギー	認証用の証明書 PrK#2	( 認証のための ) 署名操作に使用。復号できない(暗号には利用できない)、PIN によるカード保有者の認証。
	否認防止の署名用証明書 PrK#3	署名操作のみ。署名操作毎に PIN によるカード保有者の認証が必要。 否認防止の署名用証明書は、18 歳以上に発行される。
PIV 米国	PIV 認証鍵 証明書	個人識別番号 ( PIN ) を使用してカードおよびカード保有者を識別する際には、FIPS 201 で定義されている PIV 認証鍵を使用する。
	デジタル署名鍵 証明書 ( オプション )	この鍵と認証は、文書に署名するためのデジタル署名の使用をサポートす

		る。「常に PIN」というアクセスルールによって保護される。これには、鍵を使用してデジタル署名を生成するたびに、カード保有者が関与する必要がある。
	鍵管理鍵 証明書（オプション）	この鍵ペアは、鍵復元のために発行者がエスクローする。 「PIN」アクセスルールによって保護される。
公的個人 認証サービス	否認防止の署名用証明書	否認防止用の証明書のみ。

この他、IC・IDカードに格納される重要なデータとして、IC・IDカード保有者の信頼点となる認証局の自己署名証明書がある。IC・IDカード保有者は、自己署名証明書に含まれる公開鍵を信頼し、この公開鍵を使って様々な検証を行う。IC・IDカード保有者が、自分のカードを信頼するということは、ネットワークにおいては、この自己署名証明書を信頼するということに他ならない。そのためIC・IDカードに安全にカード保有者の信頼点となる認証局の自己署名証明書が格納されることは、非常に重要な意味をもつ。

以上のようなことも含めて3章では、フィンランドのFINEID、ベルギーのBELPIC、米国のPIVの3つの事例を紹介する。

#### 1.4.3. IC・IDカードに格納されるデータに対する操作

IC・IDカードには、プライベート鍵、証明書（カード保有者の証明書、カード保有者の信頼点になる証明書）などの他、IC・IDカードがカード保有者を認証するための認証情報等が格納される。IC・IDカード自体が、公開鍵暗号を用いて外部機器などの認証（外部認証）を行なための公開鍵（証明書ではなく公開鍵のみ）が格納されることもある。これらのIC・IDカードに格納されるデータに対して様々な操作（オペレーション）が行なわれる。IC・IDカードサービスに要求される機能を正確に理解するためには、これらのデータが、ICカード自体、ミドルウェア、クライアントアプリケーションにおいて、どの様に扱われるか等を理解する必要がある。表4にIC・IDカードに格

納されるデータとデータに対する操作を示す。

表 4 IC・ID カードに格納されるデータとデータに対する操作

データ	データに対する操作	説明
プライベート鍵	署名操作 (PSO: COMPUTE DIGITAL SIGNATURE) 復号 (PSO: DECIPHER)	プライベート鍵を使った操作を IC・ID カードで行なうため非常に重要。否認防止用の署名鍵の場合、復号に利用できないことも重要。
公開鍵	署名検証 (PSO: VERIFY DIGITAL SIGNATURE) 暗号 (PSO: ENCIPHER)	署名（認証も含む）目的の場合 IC・ID カード上で公開鍵の演算（暗号化）を行なう必要はない。IC・ID カード自体が外部を認証する（外部認証）の場合、カード上での署名検証が要求される。
証明書	READ BINARY GET DATA	カード保有者が、署名検証を行う場合、カードに格納された信頼点の証明書の公開鍵を利用する。また、暗号化を行なう場合、カード保有者の証明書の公開鍵を利用する。
認証情報	VERIFY RESET RETRY COUNTER	カード保有者を IC・ID カードが認証する。

IC・ID カードにとって「プライベート鍵」は、最も重要なデータである。IC・ID カード上でのプライベート鍵による「署名操作」が重要なことは、前述したとおりである。その他に重要な機能として、プライベート鍵を使ったカード上での暗号の復号がある。公開鍵暗号においては、公開鍵を使って暗号化し、プライベート鍵を使って復号する。ここで、IC・ID カード自体で利用される暗号化ではなく、クライアントアプリケーションにおける暗号化(例えば、S/MIME による暗号)は、カード保有者の公開鍵により行なわれる。これは多くの場合、カード上で行なわれる訳ではないことに注意すべきである。公開鍵を含んだ証明書が「READ BINARY」といったカードエッジ・インターフェースのコマンドを使って IC・ID カード外に取り出され、カード外で暗号化が行なわれ

る。「プライベート鍵」と「公開鍵」の性格の違いをよく理解する必要がある。表 4 の「公開鍵」は、カード管理等に利用され、カード保有者（のアプリケーション）から直接利用されない場合が多いことに注意する必要がある。

## 1.5. IC・ID カードの相互運用可能性の課題

前節では、IC・ID カードに要求される基本的な機能や格納されるデータについて説明した。こうした基本的な要求がある中、IC・ID カードの相互運用可能性の課題には何があるか、また、相互運用可能性の課題に対して、どのような対応がなされているか等について説明する。

### 1.5.1. IC・ID カードの相互運用可能性の分類

IC・ID カードを広く利用できる環境を整備するには、相互運用可能性の確保が欠かせない。そのためには、IC・ID カードの相互運用可能性の課題を正しく理解する必要がある。IC・ID カードの相互運用可能性といっても、いくつかのレイヤーが存在している。本報告書では、以下のレイヤーに分類している。この分類を説明した上で、次節以降 IC・ID カード相互運用可能性の課題を整理する。

- IC・ID カードとカードリーダー間
- カードリーダーと PC などの端末間
- カードエッジ・インターフェース
- データモデル
- 証明書の内容など PKI の相互運用

上記の分類を図 5 に図示する。

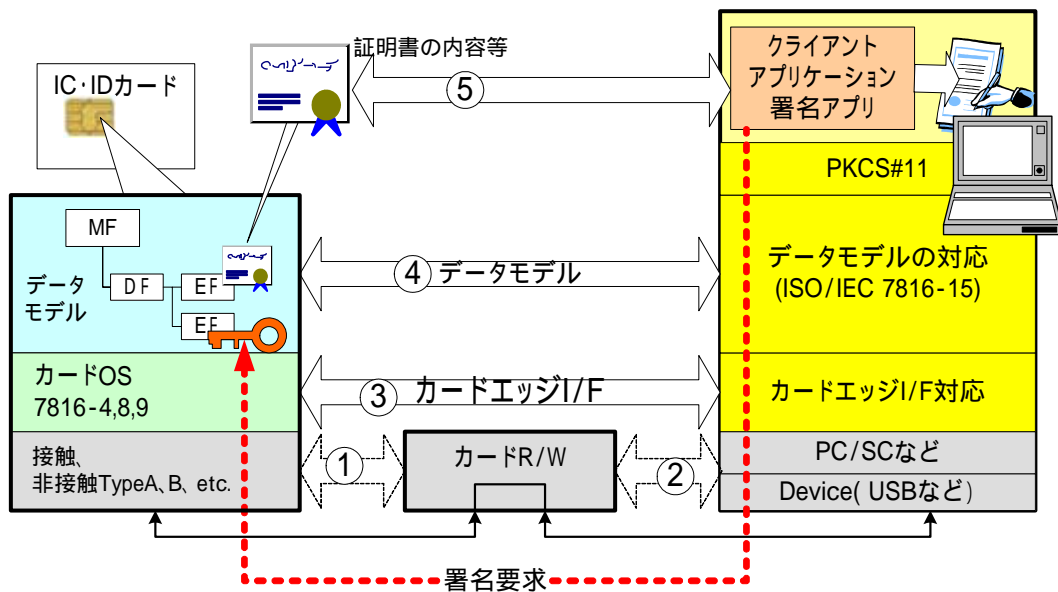


図 5 相互運用可能性の分類

次に各分類での IC・ID カードの相互運用可能性について説明する。

#### (1) IC・ID カードとカードリーダー間

IC カードのインターフェースとしては、接触型インターフェース、非接触型インターフェースがある。接触型インターフェースの標準としては、ISO/IEC 7816 シリーズの ISO/IEC7816-1<sup>[1]</sup>,2<sup>[2]</sup>,3<sup>[3]</sup>がある。また、非接触型インターフェースとしては、ISO/IEC 14443<sup>[4]</sup>[5][6][7] TYPE-A,TYPE-B があり、さらに日本国内でよく利用されている FeliCa といった規格も存在する。

相互運用性を確保するためには、IC カードとカードリーダー間において同じ規格が利用できる必要がある。複数の非接触型インターフェースをサポートしたマルチインターフェース・カードリーダーや、カード側が接触と非接触の2つのインターフェースをサポートしたデュアルインターフェースカードといったものも存在する。

#### (2) カードリーダーと PC 間

カードリーダーと端末は、USB、RS-232C、PCMCIA といったインターフェースで接続されることが多い。カードリーダーには固有のドライバーが必要になるが MS Windows (プラットフォーム)環境においては、PC/SC といった規格が存在し、カードリーダー固

有のドライバーの仕様の違いを吸収している。

### (3) カードエッジ・インターフェース

IC・ID カードと端末間の論理的なやりとり、すなわちカードエッジ・インターフェースは、ISO/IEC 7816 で定義される IC カードの基本コマンドである APDU (Application Protocol Data Unit)を使って行なわれる。この APDU を使って、IC・ID カードと PC(端末)間でデータのやりとりが行なわれるが、この APDU は、ISO/IEC 7816-4<sup>18)</sup>などによって標準化されている。しかし、実際の APDU の詳細は、IC カードの OS (カード OS) により異なるものが多く、特に、IC・ID カードで利用される複雑な「セキュリティ関連コマンド」などにおいては、差異が大きい。

IC・ID カードの端末側のアプリケーションは、多くの場合 PKI のアプリケーションである。こうしたアプリケーションは、ミドルウェアを介して利用されるが、カードエッジ・インターフェース(APDU)の差異は、こうしたミドルウェアが吸収することによって、相互運用可能性を確保している場合が多い。

### (4) データモデル

IC・ID カードでは、プライベート鍵、各種証明書、カード保有者を認証するための認証情報といったものが内部に配置される。そして、これらの情報の適正なセキュリティ確保のために、アクセス制御ルール等が設定される。こうした情報を、PKI のクライアントアプリケーションが直接扱うことは少ない。クライアントアプリケーションは、ミドルウェアを介して、これらの情報にアクセスする。この時ミドルウェアは、プライベート鍵、各種証明書、認証情報が IC・ID カード内にどのように配置されているか、また、これらの持つ各種の属性、アクセス制御ルール等を予め知っている必要がある。つまり同一のカード OS を使った IC・ID カードであっても、データモデル(ファイル構造、データ構造等)が異なれば、異なるミドルウェアが必要になることに注意すべきである。

図 6 に IC・ID カードに格納されている情報のイメージを示す。

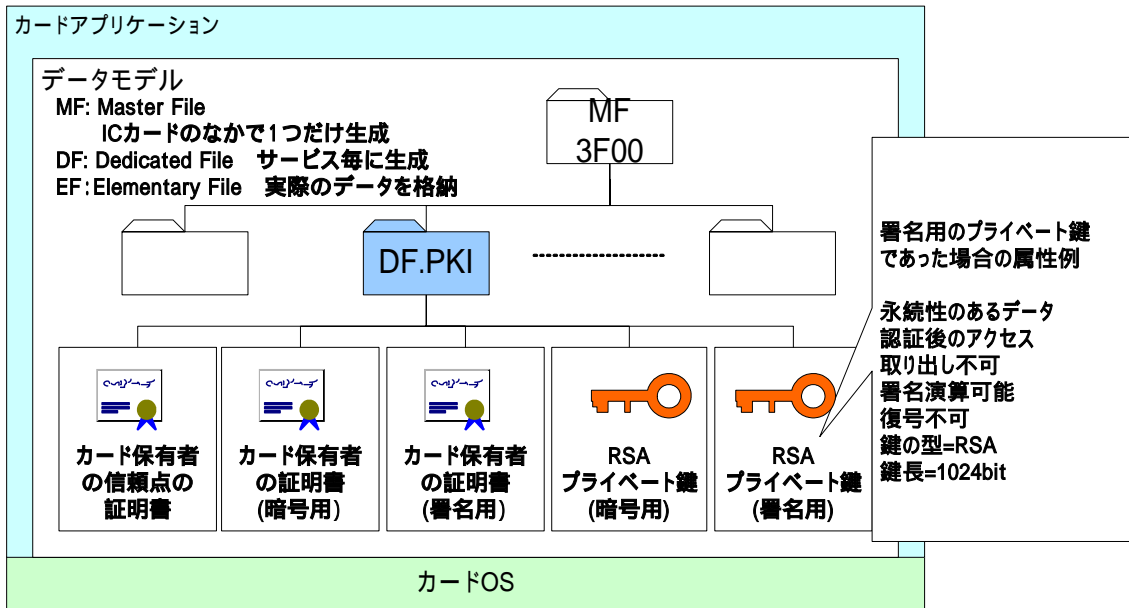


図 6 IC・ID カードに格納されている情報

(5) 証明書の内容など PKI の相互運用

IC・ID カードの鍵、証明書類が利用できたとしても、その格納された証明書も含めた相互運用可能性の確保が求められる。

以上のように IC・ID カードの相互運用可能性を 5 つに分類した。本報告書は、IC カードを使った ID カードである「IC・ID カード」の相互運用可能性が中心的なテーマである。より一般的な IC カードの相互運用可能性の問題であり比較的よく理解されている(1)、(2)の問題に関して本書では、これ以上踏み込まない。また(5)は、「IC・ID カード」というよりは、証明書発行や証明書検証を含んだ PKI の相互運用可能性の課題であり、これも、本報告書では中心的なテーマとして取り扱わない。

以上のことから本書では、(3) カードエッジ・インターフェース、(4) データモデルのふたつの相互運用可能性を主に取り上げる。

1.5.2. 相互運用可能性の課題

IC・ID カードは、既に様々なアプリケーションで利用されている。従って、相互運用可能性の問題があって利用できないというものではない。現状の IC・ID カードは、様々なクライアントアプリケーションに対応するため、IC・ID カードと PKCS#11<sup>[9]</sup>な

どのデファクトスタンダードな API をサポートしたミドルウェアが一体化したものが提供されている。図 7 のようなものが実際に多く提供されている。

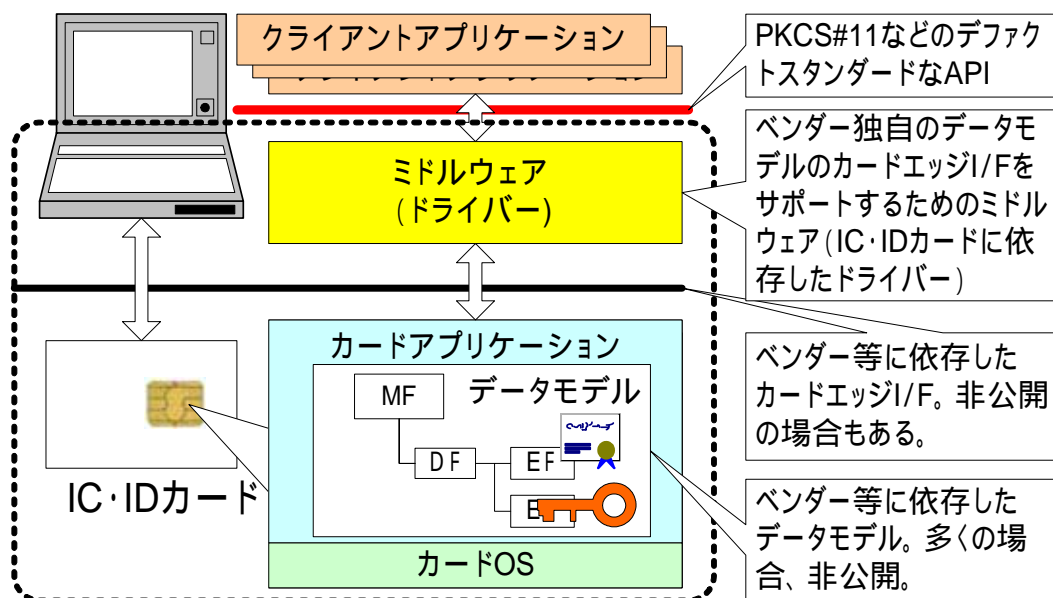


図 7 IC・IDカードの提供形態

図 7 のような IC・ID カードソリューションは、企業内等においても幅広く利用されつつある。しかし、IC・ID カードの利用の範囲を広げるためには、大きな課題がある。次にこれらの課題を説明する。

(1) PKCS#11 等デファクト標準としての API

現状よく利用されているアプリケーションインターフェース(API)は、必ずしも安定したものではない。例えば、PKCS#11 のサポートと言っても多くの場合、PKCS#11 のサブセットの実装であり、実装のレベルも様々である。また、準拠性テスト仕様 (& テストツール) の不在などの理由から、多くの問題が内在すると考えられるが、これらは「相性の問題」で片付けられてしまっているところがある。こうした問題は、少数のアプリケーションしか動作していない初期の導入時には気付かない場合もある。

(2) ベンダー独自のデータモデルとカードエッジ・インターフェースをサポートするためのミドルウェア (IC・ID カードに依存したドライバー)

IC・ID カードとミドルウェアが一体化されて提供されている。そのため、IC・ID カードとミドルウェアそれぞれの独立性がない。IC・ID カード毎のドライバーが必要になり、ドライバー同士のコンフリクト等のトラブルの元にもなっている。

(3) ベンダー等に依存したカードエッジ・インターフェース、ベンダー等に依存したデータモデル

これは、IC・ID カードとしての独立性、ポータビリティが無いことを意味する。使う環境が限定されるだけでなく、長期に保守することを困難にする可能性がある。IC・ID カードは、今後の IT 社会の基盤となるフロントエンドツールになる可能性があるが、そういった基盤となるためには、長期に渡って保守されるべきであり、たとえ、利用するプラットフォームが変化しても、移行(マイグレーション)が可能であるべきである。そのためには、IC・ID カードとしての独立性、ポータビリティは、欠かせない要件となる。

(4) マルチプラットフォームで動作する IC・ID カードが少ない

IC・ID カード自体がベンダー依存の場合、ミドルウェアの提供も限定されてしまうことが多い。この結果として、マルチプラットフォームで動作する IC・ID カードも少ない。

### 1.5.3. データモデルの課題

以上のような、IC・ID カードのカードエッジ・インターフェースやデータモデルに関連した相互運用可能性の課題は、かつて情報処理振興事業協会（現：情報処理推進機構、略称：IPA）の調査報告でも指摘されている。

「2003 年度 情報セキュリティ関連の調査に関する公募」において採択された「セキュリティ API に関する技術調査<sup>[10]</sup>」の「Part 4. IC カードなどのハードウェアトークン API」では以下のような指摘がある。

8 まとめ  
(省略)

多くの場合、CSP および PKCS #11 ライブラリは、特定の IC カードおよびファイル構造にそったものでしか扱えず、IC カード間での互換性が図れていない。そのため、このベンダーの IC カードを利用するなら、このライブラリ、逆にこのライブラリならこの IC カードというような、紐付けがなされているのが現状である。

GSC-IS の場合、そのような現状にある各々のレイヤーの互換性をすすめるために、IC カードをあつかうミドルウェアおよび、IC カードまで踏み込んだ仕様化がなされている。レイヤーの互換性の向上により、ミドルウェア開発ベンダーはビジネスチャンスを広げるために、工夫が必要となっていく。

ここで GSC-IS は、NIST( National Institute of Standards and Technology )が 2003 年に策定した Government Smart Card-Interoperability Specification Version 2.1<sup>[11]</sup> を指している。「Part 4. IC カードなどのハードウェアトークン API」では、この GSC-IS の説明も記述されている。また、GSC-IS は、本報告書の 3 章、4 章、5 章で事例研究として取り上げている米国の PIV の元になる仕様を提示している。

「2002 年度 情報セキュリティ関連の調査に関する公募」において採択された「本人認証技術の現状に関する調査<sup>[12]</sup>」では以下のような指摘がある。

### (3) ハードウェアトークンのクレデンシャルフォーマットの標準化

広く配布される IC カードなどでは、ハードウェアトークンのクレデンシャルフォーマットの標準化を考慮に入れるべきである。こうしたクレデンシャルフォーマットの標準化には、PKCS#15 ないし、7816-15 がある。現在多くのハードウェアトークンと、そのハードウェアトークンを扱うドライバーの実装は、独自のクレデンシャルフォーマットを扱う。独自のクレデンシャルフォーマットを行った場合、そのハードウェアトークンの使用は、この特定のドライバーのみ可能ということになる。これは、企業内の使用目的などの場合には大きな問題にはならない。しかし、公的な目的で、広いプラットフォーム上で使用されるためには問題になる。電子政府で使用されるハードウェアトークンには、クレデンシャルフォーマットを公開すると共に、そのフォーマットの標準化、そして、汎用的なクレデンシャルフォーマットとしての PKCS#15 (7816-15) などの採用を検討すべきである。

この記述にある PKCS#15<sup>[13]</sup> (7816-15) は、「セキュリティ API に関する技術調査」

の「Part 4. IC カードなどのハードウェアトークン API」において説明されている。PKCS#15 は、現在では、ISO/IEC において ISO/IEC 7816-15<sup>[14]</sup>という規格となっており、また、2006 年には、「IC カード - 第 15 部：暗号情報アプリケーション JIS X 6320-15<sup>[15]</sup>」として JIS 化されている。

ISO/IEC 7816-15 の考え方は、IC・ID カードのデータモデル等の「仕様」を IC・ID カード自体に格納し、ミドルウェアは接続時にこの「仕様」を読み取る。このことにより、IC・ID カードとミドルウェアが、それぞれ独立性を持ち依存性を少なくすることができる。このことは、ミドルウェアが、IC・ID カードの仕様変更（追加）に対応できることも意味する。ISO/IEC 7816-15 ではこの「仕様」を、ASN.1 構文を使ってエンコードし IC・ID カード自体に格納する。ASN.1 によるエンコードは「仕様」をコンパクトに表現し、更に、特定のマシン（IC カード、及び、端末）のアーキテクチャに依存しないポータビリティを持つことができる。この「仕様」のエンコード、及び、「仕様」を解釈するためのデコードは、IC・ID カード上で行なわれるわけでないことに注意しなければならない。

本書では、2 章で PKCS#15（7816-15）自体の説明、3 章で PKCS#15（7816-15）を取り込んだ IC・ID カードの事例として FINEID、BELPIC を説明する。また、4 章で PKCS#15 に対応したオープンソースの実装事例である OpenSC を説明している。FINEID、BELPIC 等では、ISO/IEC 7816-15 ないし、ISO/IEC 7816-15 の前身の仕様である PKCS#15 を IC・ID カードの仕様として取り入れている。ISO/IEC 7816-15 は、暗号情報アプリケーションで利用するプライベート鍵、証明書、更にこれらをアクセスするための認証情報などのファイル構造を規定している訳ではない。これらの関係を記述した情報を IC・ID カードに格納するための仕様が ISO/IEC 7816-15 である。ISO/IEC 7816-15 をサポートしたミドルウェアは、ISO/IEC 7816-15 に従って IC・ID カード上から暗号情報アプリケーションの情報を読み出し、その結果として、IC・ID カード上のプライベート鍵、証明書、認証情報などのファイル構造を知り、これらの情報を使って署名操作などの暗号情報アプリケーションの操作を行なう。

FINEID と BELPIC は、似た仕様の IC・ID カードではあるが、全く同じという訳ではない。しかし、ISO/IEC 7816-15 をサポートしたミドルウェアであれば動作させることができる。IC・ID カードに対する要件も様々であり、データモデルの仕様も様々なものが考えられる。ISO/IEC 7816-15 を仕様に取り込むことは、こうしたデータモデルの仕様の違いを吸収し、ミドルウェアを一本化することが可能となる。実際に OpenSC による FINEID と BELPIC の対応はこうしたアプローチで行われている。

図 8 に ISO/IEC 7816-15 の実装イメージを示す。ここでは、否認防止の署名に利用

するプライベート鍵に関してのみ説明している。

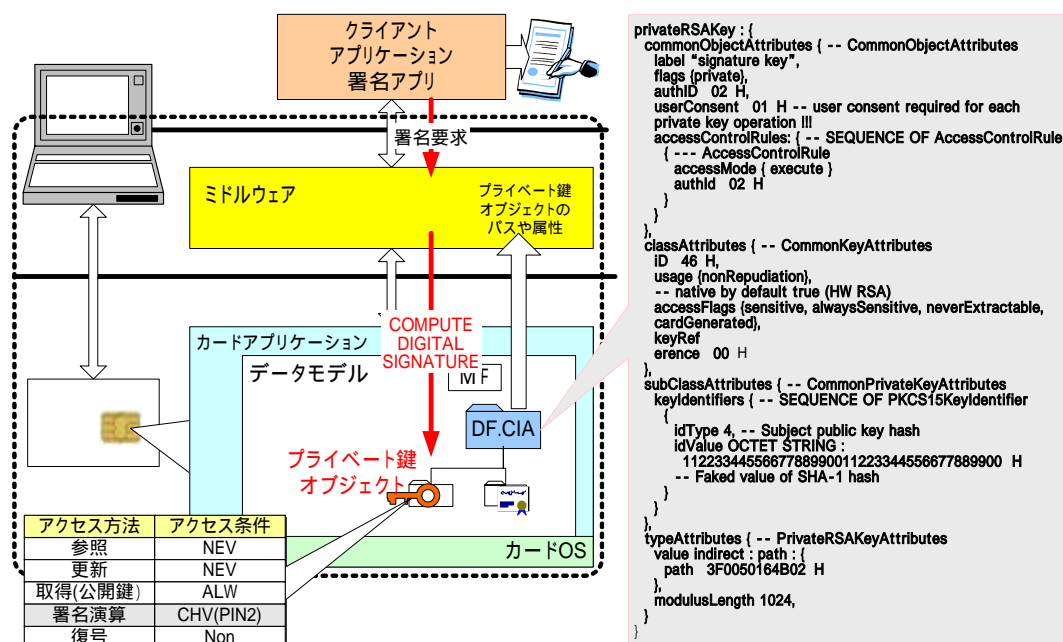


図 8 ISO/IEC 7816-15 の実装イメージ

上記のようなデータモデルに対する相互運用可能性の課題の指摘があるものの、一般的にデータモデルの標準化は、課題と見なされていない。また、ISO/IEC 7816-15 の存在、および、存在意義はあまり知られていない。これは、IC・ID カードとミドルウェアがセットになって提供される現状において、IC・ID カードの中のデータモデルには感心が低いのは当然のことと言える。

ISO/IEC 7816-15 の対応は、主にミドルウェアの対応ということになる。ISO/IEC 7816-15 が定義している仕様の範囲は非常に広いため、ISO/IEC 7816-15 に従うと言うだけでは、実際の中ウェアの実装や相互運用可能性の確保は難しい。ISO/IEC 7816-15 を採用するためには、プロファイルの策定が必要であり、プロファイル抜きに相互運用可能性の確保は難しい。本報告書では、ISO/IEC 7816-15 の前身の仕様である PKCS#15 を採用した IC・ID カードである FINEID、BELPC の仕様や、PKCS#15 に対応したミドルウェアである、OpenSC の実装などから、このプロファイル案を導き出すために必要な調査を行なう。

#### 1.5.4. カードエッジ・インターフェースの課題

データモデルの標準化だけでは、IC・ID カードのポータビリティは達成できない。

やはり、カードエッジ・インターフェースの問題をクリアする必要がある。しかし、過去の経緯などから、カードエッジ・インターフェースの標準化の対応は、なかなか難しい問題を抱えているのかもしれない。現在のところ、一般に販売されている IC・ID カードのカードエッジ・インターフェースは、標準に対応していないものが多いのが現状のようである。ISO/IEC 7816-4:2004、日本国内で策定した JICSAP 2.0 等の実装は、必ずしも進んでいないように見受けられる。

このようなことの原因としては、これまでのところ、幅広い利用範囲が必要な分野において PKI 等を利用する要求が必ずしも多くないこと、クライアントアプリケーション・インターフェースで吸収するという考えで IC・ID カードは提供されていること等が挙げられる。

標準と言われるものの複雑さも原因のひとつかもしれない。複雑さの対応には、コストの問題がある。また、複雑すぎる標準に対しては、相互運用可能性を確保しようとするインセンティブが働きにくい。

標準的なカードエッジ・インターフェースが利用されていない状況においては、IC・ID カードのポータビリティは確保できない。現時点において、クライアントアプリケーション・インターフェースレベルでの相互運用可能性確保で問題がなくても、IC・ID がカード広く利用されるほど、そのポータビリティの確保が重要になるということが正しく認識されるべきである。

#### 1.5.5. ミドルウェアの重要性

「モノとしての IC カード」にしか関心を持たない者に対して、ユーザインターフェース等でもないミドルウェアの実体を説明するのは難しく、また「見える化」も困難なところがある。しかし、複雑なセキュリティ要件や、様々なプロトコル、そして信頼関係等の大多数を実装しているのはミドルウェアであり、その重要性は十分に認識されるべきである。IC・ID カード、および、IC・ID カードサービスにおいても、複雑な「セキュリティ関連コマンド」等を扱うためには、ミドルウェアの存在は必須のものと言える。また、CPU やメモリ等のリソースの限られた IC・ID カードではなく、リソースが豊富な端末やサーバ側に多くの複雑な機能を配置するのは当然とも言える。これまでの IC カード単体に閉じた標準化は、システム全体のリソース配分をあるべき姿にすることが難しく、結果として IC カード自体に多くの機能を詰め込むきらいがあったのではないだろうか。リソースの限られた IC カードを使って、ミドルウェアを含めたセキュアで適切なコストの IC・ID カードサービスの実装を検討する必要がある。こうしたこと

から、端末側のミドルウェアの持つ機能や、マルチプラットフォーム対応などが重要な意味を持つことが多い。実際、IC・ID カードの調達において、ミドルウェアが対応できるプラットフォームが購入の決め手になっている場合もある。

IC・ID カード自体以上に、ミドルウェアの重要性が認識されるべきであるが、IC・ID カードとミドルウェアが一体化して提供されているため、一般にミドルウェア自体の重要性はあまり認識されていない。過去には、IC・ID カードのためのミドルウェアの開発コストが過少評価されていた可能性がある。4 章ではミドルウェアの実装の事例を取り上げているが、ここで紹介している OpenSC の場合、汎用的なミドルウェアとなっており、また、様々なユーティリティを含んではいるが、9 万ステップものソースコードから構成されている。これは、カードリーダーに対する端末のドライバーの部分のソースコードを含んでいない。

本報告書では、4 章でオープンソースの IC・ID カードのミドルウェアの開発プロジェクトである OpenSC を取り上げている。現在のところ IC カードに関連したオープンソースプロジェクトは少ない。また、日本国内に関しては、皆無に近い。PKI に利用する IC カードということで敷居が高いが、OpenSC などのオープンソースは、IC・ID カードサービス全体のアーキテクチャを理解する上で非常に役立つ。実際の実装を見ることにより、相互運用可能性の問題も見えてくる面がある。

相互運用可能性の課題を理解する上で、IC・ID カードの内の「データモデル」を含んだ、ミドルウェアのコスト、また、IC・ID カードサービスとしてのアーキテクチャが理解されるべきである。こうしたことは、PKI 等のソフトウェアの開発者の間でもあまり知られていない。この「ミドルウェア」を利用する開発者は多数存在しており、彼らにはアプリケーション・インタフェースの必然性はよく理解されている。このため PKCS#11、CryptoAPI などの API は、知られているが、PKCS#15 や ISO/IEC 7816-15 は、あまり知られていない。

標準化に関しては、IC・ID カードのミドルウェアを中心に据えた ISO/IEC 24727<sup>[16][17][18][19][20]</sup>の策定が進んでいる。IC・ID カード、IC・ID カードサービスに対するミドルウェアという考えは、ISO/IEC 24727 によって示された。これまで、ミドルウェアに相当するものは、カードドライバーという表現がなされ、これは、カードに付属した、また、カードに依存したドライバーという考えに基づいている。それに対してミドルウェアは、独立したコンポーネントとしての意味合がある。ISO/IEC 24727 は、IC・ID カードサービスとしてのアーキテクチャの整備であり、また、標準化の動きでもある。

本報告書では、米国の PIV の事例を取り上げている。PIV の仕様に関しては、NIST

が策定した SP800-73<sup>[21]</sup>中に、カードエッジ・インターフェース、クライアントアプリケーション・インターフェース、データモデルの仕様が提示されている。この規定によって、PIV では、IC・ID カードとミドルウェアが完全に分離しており、それぞれ独立したコンポーネントとして調達可能な枠組みを提供している。これは、ミドルウェアが IC・ID カードのオマケではないことを示している。

## 1.6. IC・ID カードの実装、展開

広く配布され、そして様々な場面で利用することを目的とした IC・ID カードの場合、カードエッジ・インターフェースとデータモデルが相互運用可能性の大きな課題であることを説明した。広く配布され、様々な場面で利用することを想定した IC・ID カードのプロジェクトは世界各国で進んでいる。報告書では、こうした事例としてフィンランドの FINEID、ベルギーの BELPIC、米国の PIV を取り上げている。これらのプロジェクトの共通点は、相互運用可能性に関する課題に積極的に取り組んでいることであり、そのため、仕様が非常にオープンであるということが言えるであろう。この中で FINEID 自体は、普及に成功している訳でない。しかし、この FINEID の仕様は、欧州の eID の仕様に大きな影響を及ぼしている。BELPIC は、12 歳以上の全国民への配布を行なっている途中であり、2006 年 10 月現在において 400 万枚以上のカードが発行されている<sup>[22]</sup>。今後は、利用面からの動向が注目される。

現時点で、最も注目すべき動向は米国の PIV であろう。ホームランド・セキュリティ大統領指令 12 (HSPD-12<sup>[23]</sup>) (連邦政府職員と契約業者の共通識別基準のためのポリシー) を最上位のポリシーとしてトップダウンな展開がなされている。PIV は、PIV(カードとミドルウェア)の仕様、PIV のテスト仕様、PIV のレファレンス実装、PIV のテストツールなどがセットで広く公開され提供されている。更に PIV の準拠性に対する認定制度が立ち上がっている。これらは、IC・ID カードの相互運用可能性を確保するためのフレームワークと言える。

表 5 に、FINEID、BELPIC、PIV のそれぞれの説明と、本報告書での説明の意図を示す。

表 5 本書で説明する IC・ID カード

IC・ID カード	説明	報告書の意図
FINEID フィンランド	FINEID は、フィンランド国民 IC カードであり、1999 年末から発行されている。ISO/IEC 7816-15 に準拠している。	PKCS#15 に準拠したカードの事例としての説明。IC・ID カードのとしてオープンで簡潔な仕様を提示している。ミドルウェアとして OpenSC が利用できる。
BELPIC ベルギー	BELPIC( Belgian Personal Identity Card)は、ベルギー国民 IC カード ベルギー全土への全面的導入は 2004 年にスタートしている。	PKCS#15 に準拠したカードの事例としての説明。OpenSC を中心にミドルウェアを構成しており、多くの BELPIC をサポートするソフトウェア(ミドルウェアとアプリケーション)のソースコードが公開されている。
PIV 米国	米国政府全体を対象とするセキュアで信頼性のある身分証標準を規定する大統領指令 (HSPD-12) に基づき、各政府機関は、NIST が策定した FIPS-201 に準拠するカードの発行が義務付けられている。2006 年 10 月 27 日までに、「PIV-II (IC カードベースの相互運用可能な個人 ID 認証システム技術仕様)」に則った身分証の発行を開始することが要求されている。	大量のドキュメントが公開されている。 ミドルウェア(PIV ミドルウェア)のレファレンス実装のソースコードが公開されている。 PIV カードアプリケーション、PIV ミドルウェアの認定制度がある。 これらを事例として説明する。

このような IC・ID カードのプロジェクトの課題は、もちろん IC・ID カードの相互運用可能性の確保だけではない。単に IC・ID カードと言うよりは、「公共、公的、ないし公的に近い分野（例えば医療、福祉分野）の広域におけるアイデンティティ管理をいかに行なうか」といったところが焦点になる。これには、プライバシー問題、セキュリティ

ティ問題、これらにかかるコストの負担をどこに求めるか等、検討すべき課題は多い。

IC・ID カードに関連した取り組みという点においても、プライバシー問題、セキュリティ問題は避けて通れない。現状、日本国内における取り組みは、必ずしもオープンとは言えず、これらを議論するだけの情報が公開されているようには見えない。プライバシー問題、セキュリティ問題の解決にしても、仕様の透明性と技術の理解が必要になる。ブラックボックスでは、これらの解決にはならない。本調査報告書の FINEID、BELPIC、PIV の例にしても、これらは全て公開された情報に基づいて記述されている。少なくとも我が国の公的個人認証サービス、国家公務員証 IC カード等では、ここまでの情報は公開されておらず、これらの仕様の入手などには機密保持契約の締結などが必要になる。これらは、セキュリティ上の理由で非公開と説明されることが多い。仕様の非公開は、セキュリティ上プラスになるのか、きちんとした議論がなされるべきである。

仕様の非公開は、相互運用可能性を確保する上でも大きな障害になる可能性がある。先にも説明したとおり、相互運用可能性を確保のためには、様々な関係者の理解が必要になるが、機密保持契約が必要な仕様は、幅広い議論を行なうことを実質的に困難にしている。現状、こうしたことがあまり認識されていないのは、相互運用可能性の範囲を非常に狭く捉えているからだと思われる。

4 章で説明する IC・ID カードサービスのミドルウェアのオープンソフトプロジェクトである OpenSC では、FINEID、BELPIC、PIV のいずれもサポートしている。IC・ID カードに限らず、複雑な標準の実装においては、レファレンス実装となり得る公開されたソースコードの果す意義は大きい。実際 OpenSC も欧州の eID の相互運用可能性の問題の解決に対して大きな影響を与えているように見受けられる。OpenSC は、非常に多くの IC カード、IC・ID カードをサポートしているが、これらは、仕様が開示されて初めて可能になる。

5 章では、米国の PIV の背景を説明しているが、PIV の仕様である FIPS-201、SP800-73 と言った技術文書は原案段階においても公開されている。そして、この原案に対して 80 を超える個人と組織から 1900 以上のパブリックコメントが寄せられている (<http://csrc.nist.gov/piv-program/FIPS201-Public-Comments.html>)。こうしたことは、PIV の相互運用可能性の課題を解決し、PIV が広く受け入れられる土壌を作ることなのでないだろうか。

IC・ID カードは、中長期的なライフサイクル管理とセキュリティが必要だということも理解する必要がある。IC・ID カードとして独立性、ポータビリティがないとすると、使う環境が限定されるだけでなく、長期に渡り保守することを困難にする。IC・ID カードを今後の IT 社会の基盤となるフロントエンドツールとすることを目標とするな

らば、現時点での相互運用可能性の問題をクリアするだけでなく、将来に渡ってマイグレーションが可能なフレームワークも検討する必要がある。中長期的なセキュリティの確保のためには、暗号アルゴリズムのマイグレーションも考慮される必要がある。米国の PIV の仕様書類に SP800-78<sup>[24]</sup> 「PIV のための暗号アルゴリズムと鍵サイズ」がある。この SP800-78 では、ハッシュアルゴリズムの SHA-1 から SHA-2 ファミリへの移行や、RSA 1024bit から 2048bit 等への移行スケジュールが示されている。

IC・ID カードの実装、展開には、マルチセクター、マルチベンダー、マルチプラットフォームと利用範囲、応用範囲等、従来、考えられていたドメインの枠を超えた幅広い相互運用可能性の確保が欠かせないが、こうした相互運用可能性の確保の問題解決については、様々な関係者による、幅広い標準技術や実装技術等の適切な理解と、積極的な取り組みが必要になる。そのためには十分な情報が公開される必要がある。

## 2. 標準化動向に関する調査

### 2.1. IC カードサービスの概要

IC カードは、カード上に処理機能やデータ保持領域を持ち、それらを使いなんらかの計算実行やデータ保持を行う。カード上に存在するそれらの機能セットをカードアプリケーションと呼ぶ。一方、カードアプリケーションにアクセスする処理ソフトウェアをクライアントアプリケーションと呼ぶ。

実際にカードにアクセスするには、カードに対するリーダ・ライタが存在し、それらの中で電気信号が交換される。さらにその電気信号に基づいて、リーダ・ライタと接続されているデバイス（PC の場合、USB など）間での信号交換、さらにデバイスを管理している機器で情報を受け渡していく。それらの過程において、そもそもの電気信号はより論理的な情報へと変換されていく。そのため、実際にカードアプリケーションに対してクライアントアプリケーションがアクセスする際には、そもそもの電気信号に関する理解はなくてもよい。カードアプリケーションとクライアントアプリケーションの間に、中間層としてミドルウェアやドライバが存在し、それらの間を仲介する。図 9 では、クライアントアプリケーションと IC カード間にミドルウェア層が存在し、情報を仲介する様子を示している。

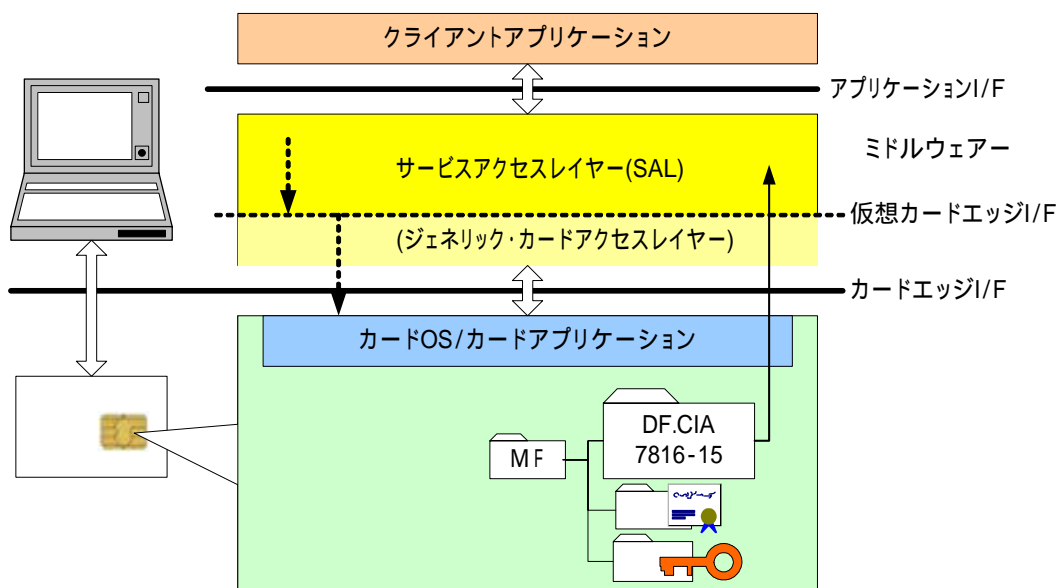


図 9 IC カードにおける情報伝達のレイヤー

一般的にクライアントアプリケーションは、ミドルウェアにより与えられた API を利用しており、それを通じてカードアプリケーションにアクセスする。そのためクライアントアプリケーションは、実際の電気信号やカードアプリケーションが持つ処理機能やデータ保持の機構などを意識していない。そこで本報告書では、クライアントアプリケーションがアクセスするインターフェース（クライアントアプリケーションインターフェース）から実際のカードアプリケーション、さらに IC カード自身を含んだ全体を「IC・ID カードサービス」と呼ぶこととする。

実際に IC カードサービスという言い方は、ISO/IEC で現在策定中である標準の ISO/IEC 24727 「ID カード - IC カードプログラミングインターフェース (Identification cards -- Integrated circuit card programming interfaces)」シリーズの ISO/IEC 24727-1<sup>[16]</sup>において定義されている。

ISO/IEC 24727 シリーズは、多様なアプリケーションドメインにまたがった相互運用可能性を望むようなクライアントアプリケーションを実現するときに利用される標準として策定中のものである。カードアプリケーションとクライアントアプリケーション間で情報やトランザクションをやりとりするにあたってのプログラミングインターフェースを提供している。2006 年 10 月現在、24727 シリーズは下記の 5 つより構成されている。

- ISO/IEC FDIS 24727-1 アーキテクチャ
- ISO/IEC FCD 24727-2<sup>[17]</sup> 汎用カードインターフェース
- ISO/IEC CD 24727-3<sup>[18]</sup> アプリケーションインターフェース
- ISO/IEC NP 24727-4<sup>[19]</sup> API 管理 ( API administration )
- ISO/IEC NP 24727-5<sup>[20]</sup> テスト

24727-1 は、24727 全体の序説となるパートであり、概念的なフレームワークの説明を行っている。これを元に、他のパートはこの概念の技術的詳細を提供している。24727-1 については 2.1.1.1 節において解説を行う。24727-2 は、カードコマンドレベルでのプログラミングインターフェースを定義するものであり、2.1.2 節で示す ISO/IEC 7816 標準群のコンセプトやデータ構造、カードコマンドを具現化するためのものである。24727-2 は 2.4.7 節であらためて解説する。24727-3 は、言語や実装と独立してアプリケーションレベルでのインターフェースを定義するものである。こちらも 2.6.1 節であらためて解説する。

### 2.1.1. ISO/IEC 24727-1

ISO/IEC 24727-1 は 2006 年 11 月の時点ではまだ国際規格として成立してはならず、最終国際規格案（FDIS）の段階であり、最終的な承認を経た後に国際規格として発行される予定となっている。

24727-1 では、IC カードとそれを利用する環境の機能性を分割してそのアーキテクチャを示している。まず、大きくクライアントアプリケーションと、それにより利用可能なサービスに分割される。さらに、利用可能サービスは階層化して考えられており、サービスインターフェース、汎用カードインターフェース、IC カード上のカードアプリケーションと分割される。図 10 は 24727 全体のアーキテクチャを示す図である。

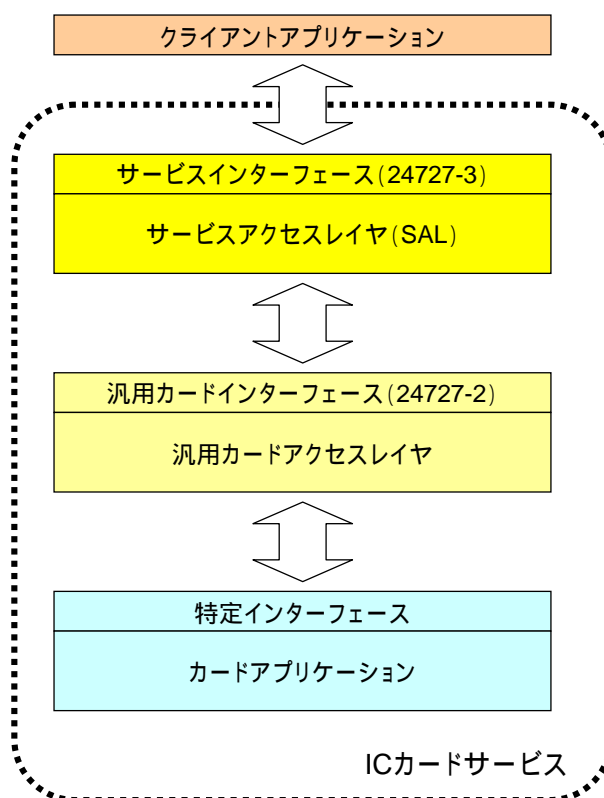


図 10 ISO/IEC 24727 のアーキテクチャ

このアーキテクチャに基づいて、24727-2 や 24727-3 では詳細な仕様を規定している。アーキテクチャを実装する際には、IC カードサービスの中で IC カード自身がどの部分の機能までを持つか、あるいはミドルウェアがどの機能を実現するか、によりさまざまな形態が考えられる。24727-1 では付録としていくつかの例を示している。基本的な物理構成として、IC カードサービスを実現するコンポーネントが、IC カード自身が持つ

コンポーネントと IC カード以外が持つコンポーネントに分けられる。図 11 に物理アーキテクチャを示す。

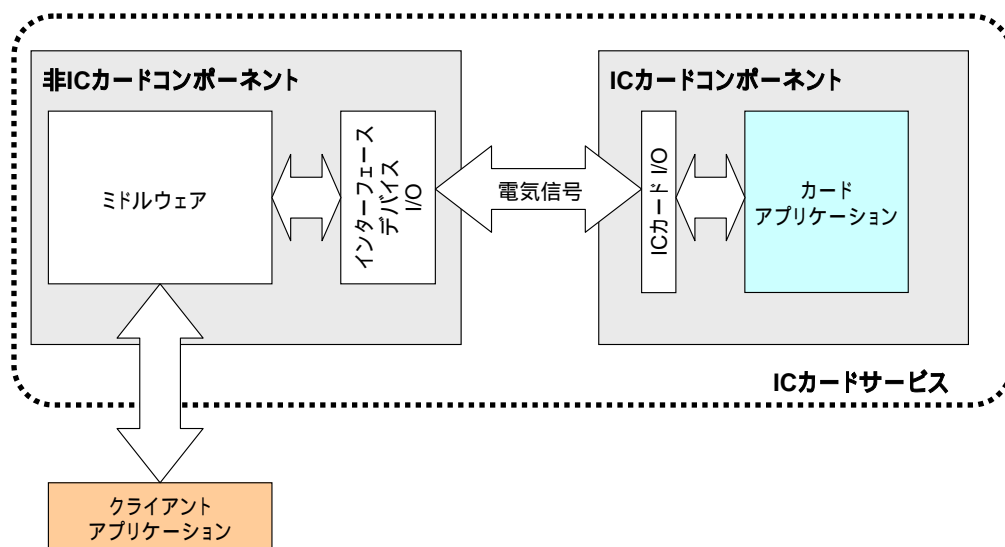


図 11 IC カードサービスの物理アーキテクチャ

図 12 では、汎用カードアクセスレイヤーとサービスアクセスレイヤーがそれぞれ別のミドルウェアで実現されている。米国国立標準技術研究所（NIST）が米国連邦政府向けの IC カードの相互運用性を図るために作成した「Government SmartCard Interoperability Specification（GSC-IS）<sup>[11]</sup>」はこのモデルにあたる。また、図 13 は汎用カードアクセスレイヤーが IC カード側にある例を示している。3.4 節で解説する米国連邦政府職員向け ID カードプロジェクトである PIV（個人識別情報の検証（Personal Identity Verification））はこのモデルにあたる。

さらに図 14 では 24727 に準拠していないアプリケーションと混在する環境での例を示している。これらの図中で示されている CCD（Card Capability Description）は IC カード内に 1 つだけ存在し IC カードが持つ機能とカードアプリケーションを示すものである。CCD は、IC カード内にあるカードアプリケーションの AID を示すデータオブジェクトを含むことが可能であり、複数のカードアプリケーションの識別を可能にしている。また ACD（Application Capability Description）はカードアプリケーションが持つ機能を示すものであり、それぞれのカードアプリケーションごとに存在する。CCD、ACD とともにデータオブジェクトとして手続き型要素（Procedural Elements）を持つ。手続き型要素は、汎用カードインターフェースへのアクセスを特定インターフェースのアクセスへと変換する方法を示したデータである。

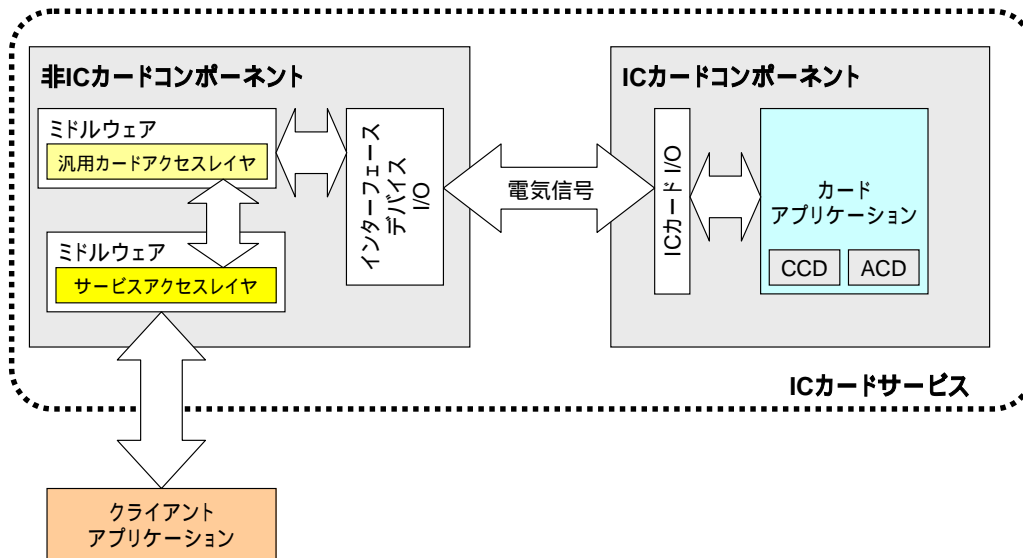


図 12 アーキテクチャ例 1 : 2つのミドルウェア

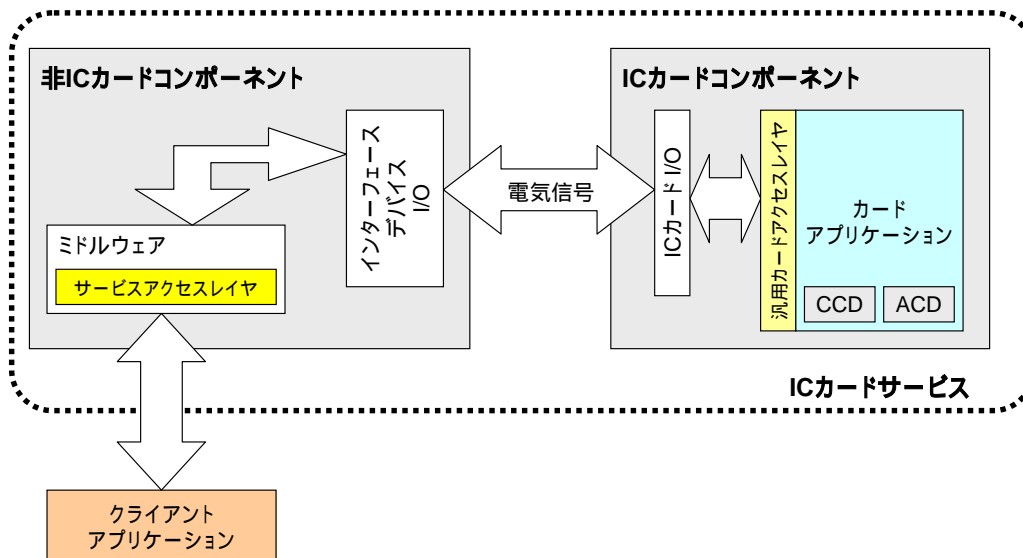


図 13 アーキテクチャ例 2 : IC カード上の汎用カードアクセスレイヤー

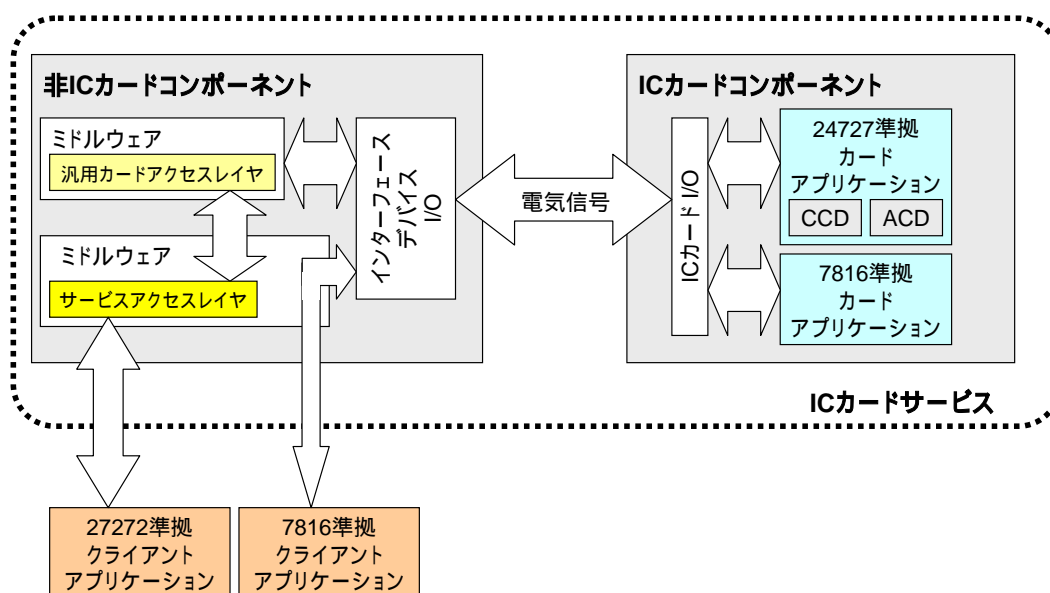


図 14 アーキテクチャ例 3 : ISO/IEC 24727 準拠/非準拠クライアントアプリケーションの混在

ISO/IEC 24727 シリーズで規定している標準は、IC カードにおけるプログラミングインターフェースを規定したもの、とされているが、内容として IC・ID カードに必要なアーキテクチャやコマンド・データモデルなどが標準化されているように思われる。

### 2.1.2. ISO/IEC 7816

2.1.1 節で触れた ISO/IEC 24727 シリーズは 2006 年 11 月の時点でまだ策定中のものであり、また IC カード用のプログラミングインターフェースを規定するものとなっている。ISO/IEC における IC カード関連の標準として代表的なものに ISO/IEC 7816「ID カード - IC カード」シリーズがある。7816 シリーズは、IC カード自体の規定とその利用法に関する標準である。5 つのパートが電氣的な接触、つまり接触カードに関するものである。下記にその名称を示す。

- ISO/IEC 7816-1<sup>[1]</sup> 接触カードの物理的特性
- ISO/IEC 7816-2<sup>[2]</sup> 接触端子の寸法と位置
- ISO/IEC 7816-3<sup>[3]</sup> 非同期カードにおける電氣的インターフェースと伝送プロトコル
- ISO/IEC 7816-10<sup>[25]</sup> 同期カードにおける電氣的インターフェースとリセット応答 (ATR)

- ISO/IEC 7816-12<sup>[26]</sup> USB カードにおける電氣的インターフェースと操作手続き

他のパートは、接触・非接触を問わず IC カードに共通するものとなっている。下記にその名称を示す。

- ISO/IEC 7816-4<sup>[8]</sup> 構成・セキュリティ・交換のためのコマンド
- ISO/IEC 7816-5<sup>[27]</sup> アプリケーションプロバイダの登録
- ISO/IEC 7816-7<sup>[28]</sup> SCQL ( Structured Card Query Language ) 用コマンド
- ISO/IEC 7816-8<sup>[29]</sup> セキュリティオペレーション用コマンド
- ISO/IEC 7816-9<sup>[30]</sup> カード管理用コマンド
- ISO/IEC 7816-11<sup>[31]</sup> バイオメトリック手法を通じた個人認証
- ISO/IEC 7816-15<sup>[14]</sup> 暗号情報アプリケーション

ISO/IEC 7816-4、7、8、9 はそれぞれ IC カードに対して入力されるカードコマンドを規定したものとなっているが、これらは 2004 年から 2005 年にかけて大きく改訂されたものである。改訂前では、IC カードの構成やセキュリティとして重要な概念などが分散している傾向にあった。IC カードの標準を策定していくにあたり、順次標準化されていったための結果である。改訂にあたりそれらの構成やセキュリティの概念、さらにカードコマンドの基本的な部分についてをすべて 7816-4 にまとめ、その上で用途別のカードコマンドを規定するものとして 7816-7、8、9 をあらためて記した。そのため、改訂前とはそれぞれのタイトルも変化していることに注意されたい。

本章では、IC・ID カードの相互運用可能性に関連する項目として特にカードエッジ・インターフェースとデータモデルに注目し、それらに関連する標準化や現状を解説する。2.2 節では IC カードに搭載される OS について解説し、2.3 節では IC カードにおけるセキュリティ保持の仕組みを解説する。2.4 節ではカードエッジ・インターフェースの標準化に関して、相互運用可能性に関連する項目を抽出し解説する。また 2.5 節では暗号情報などを保持するデータモデルについて触れる。最後に 2.6 節でカードサービスの API の標準化動向について解説する。

## 2.2. IC カード OS

IC カードは OS を動作させ、その上でカードアプリケーションを動かす。IC カードの OS にはいくつかの種類があり、それらは大きく専用 OS と呼ばれるものとマルチア

アプリケーション OS と呼ばれるものの 2 つに分けられる。

専用 OS と呼ばれるものは、特定のカードアプリケーションを利用するための OS であり、カードアプリケーションと OS の機能がともにカードの書き換え不可能な領域に最初から書かれている。ネイティブ OS とも言われる。

一方で、近年ではカードアプリケーションの追加などが可能になる形態として、ハードウェアとしてのカードに特有な OS と、ハードウェアの差異を吸収する仮想マシン (Virtual Machine) を用意し、それらを書き換え不可能な領域に書き込んでおき、書き換えが可能な領域にカードアプリケーションを搭載するものがある。こういった仮想マシンを搭載したものをマルチアプリケーション OS と呼んでいる。代表的なものとして、Java の仮想マシンを搭載した Java カードや、MULTOS カードがある。マルチアプリケーション OS では、ハードウェアの差異を吸収することが可能なため、さまざまなカードアプリケーションがハードウェアを問わずに利用することが期待される一方で、カードアプリケーションを利用する際に仮想マシンを介しての実行となるためにその速度が遅くなることがある。

特に注目すべき点として、GSC-IS や PIV ではその仕様がカード OS を規定していないことがある。実際に PIV で認定された IC カードでは、OS として Java や MULTOS をつかうものもあれば、ネイティブ OS を使うものもある。GSC-IS や PIV ではカード OS を隠蔽し、カードとミドルウェアの境界となるカードエッジインタフェースを規定しているため、ミドルウェアはカード OS に依存しないようになっている。これによって PIV ではカード OS に依存するカードアプリケーションと、カードエッジインタフェースに依存するミドルウェアで別々の認定を与えることを可能としている。

### 2.3. セキュリティ保持の仕組み

IC カード上のセキュリティは、物理的には耐タンパー性を持った IC チップにさまざまな情報を入れることで実現される。一方、IC チップ内部のデータを扱うときにもそれらのデータに関するセキュリティが重要であり、ISO/IEC7816-4 ではそういった論理的なセキュリティを実現するためのいくつかの仕組みを規定している。本節ではこれらのうち、アプリケーション識別子、セキュリティ属性、セキュリティ環境について解説する。

### 2.3.1. アプリケーション識別子 (AID)

IC カード内にあるカードアプリケーションが複数ある場合のアプリケーション選択や、そのアプリケーションのユニーク性を持たせるためのものとして、アプリケーション識別子 (Application Identifier : AID) がある。クライアント側は AID を見ることでその動作を決める。AID は ISO/IEC 7816-4 で規定されている。

AID は最大 16 バイトのデータより構成されるものである。最初のバイトの上位 4 ビットによりカテゴリが定められている。そのカテゴリを表 6 に示す。

表 6 AID カテゴリ

値	カテゴリ	カテゴリの説明
0-9		ISO/7812-1 との互換性を持たせるために予約
A	国際 (International)	ISO/7816-5 にもとづいたアプリケーション提供者の国際登録
B,C		将来のために予約
D	国内 (National)	ISO/IEC7816-5 にもとづいたアプリケーション提供者の国内登録
E	標準 (Standard)	ISO/IEC 8825-1 にもとづいた OID による標準の識別
F	独自 (Proprietary)	アプリケーション提供者の登録なし

AID の最初のバイトの上位 4 ビットが 0xA で表される国際登録の AID では、最初の 5 バイトがアプリケーション提供者を示す識別子 (Registered Application Provider Identifier : RID) であり、続くバイトで提供者が規定するアプリケーション識別子 (Proprietary Application Identifier Extension : PIX) を示しており、RID と PIX を組み合わせることで AID のユニーク性を実現している (図 15)。RID の登録手順は ISO/IEC 7816-5 で規定されている。

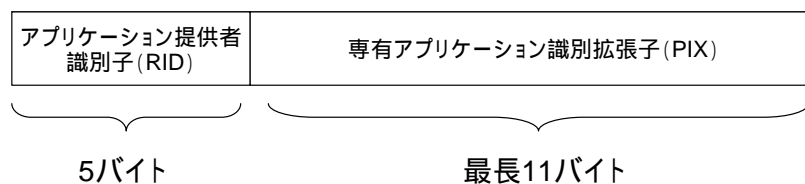


図 15 AID のデータ構成

AID の最初のバイトの上位 4 ビットが 0xE で表される場合、それは ISO/IEC の標準で規定されたアプリケーションであることを示している。本報告書に関連する AID の一覧を表 7 に示す。

表 7 本報告書に関連する AID 一覧

仕様名	AID
ISO/IEC 7816 - 15	E8 28 BD 08 0F
PKCS#15	A0 00 00 00 63 50 4B 43 53 2D 31 35
ESIGN	A0 00 00 01 67 45 53 49 47 4E
FINEID	A0 00 00 00 63 50 4B 43 53 2D 31 35
BELPIC	A0 00 00 01 77 50 4B 43 53 2D 31 35
PIV	A0 00 00 03 08 00 00 10 00 01

ISO/IEC 7816-15 においては最初の 5 バイトが「E8 28 BD 08 0F」で示されることが規定されている。一方 ISO/IEC 7816-15 の基となっている PKCS#15 では AID が「A0 00 00 00 63 50 4B 43 53 2D 31 35」となっている。後半の 7 バイトは ASCII コードで「PKCS-15」を示しており、前半の 5 バイトが RID となっていることがわかる。この AID は 7816-15 においても利用可能であることが示されている。また事例として 3 章で解説している FINEID でも PKCS#15 の AID を利用している。一方で同じく 3 章で解説している BELPIC では「A0 00 00 01 77 50 4B 43 53 2D 31 35」を利用している。前半の 5 バイト (RID 部) は異なっているが、後半の 7 バイト (PIX) が PKCS#15 の AID と同じく ASCII コードで「PKCS-15」を示すものとなっていることに注目したい。米国の PIV では「A0 00 00 03 08 00 00 10 00 01 00」が利用されており、RID 部は PIV の標準を策定した NIST のものとなっている。

### 2.3.2. セキュリティ属性

IC カード内のデータは基本的にファイルという形式を取っており、それらに対してアクセスしデータを取得したり、内部のファイルを用いた計算を IC カードで行ったりしている。ファイルの種類は専用ファイル (Dedicated File : DF) と基礎ファイル (Elementary File : EF) に分かれている。DF はコンピュータのファイルシステムで言えばディレクトリやフォルダにあたるものであり、EF はコンピュータファイルシステム上のファイルにあたる。

ファイルを特定するにはいくつかの方法があるが、ここではパスによる指定を解説する。EF や DF はそれぞれ識別子を持ち、ある DF 下にある EF 群は DF 内で一意に識別可能なように識別子が与えられている。IC カード内のファイル構造は、マスターファイル (MF) を頂点とした木構造になっている。ファイル指定を行うにあたり、MF から指定する対象までの識別子を羅列するものを絶対パスと言い、現在の位置からの相対的な位置を表現する方法を相対パスという。

ファイルにアクセスするために、まずどのファイルにアクセスするかを決定しなければならない。そこで IC カードにファイル選択のカードコマンドを発行する。その後、該当ファイルに対してさまざまな処理を行うのであるが、ファイル選択のカードコマンド発行 (SELECT コマンド) を行ったときの IC カードからの応答であるファイル管理情報 (File Control Information : FCI) に該当ファイルはどういった利用が可能か、といった情報が含まれている。FCI の中に含まれるこれらの情報をセキュリティ属性 (Security Attribute) と言う。ファイル選択によるセキュリティ属性の通知の概念図を図 16 に示す。

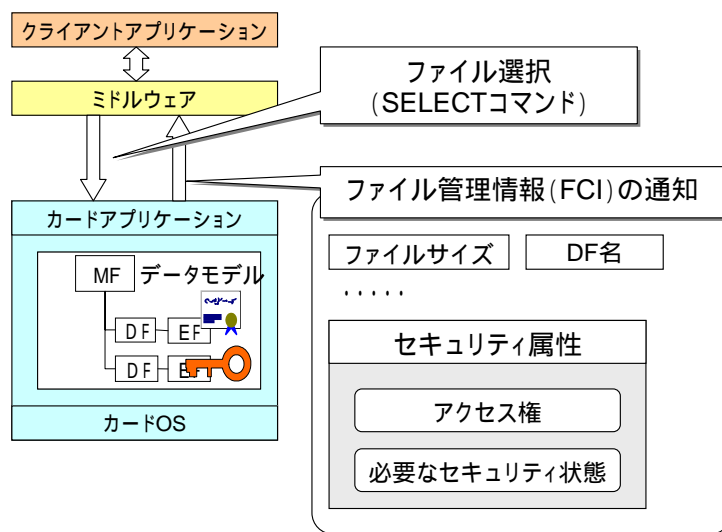


図 16 セキュリティ属性の通知

セキュリティ属性は ISO/IEC 7816-4 で定義されているものであり、もっとも基本的な Compact Format では、そのファイルへのアクセスルールが示されている。アクセスルールは、そのファイルへのアクセス制御を示すアクセスモードバイトと、アクセスに必要なとなるセキュリティ状態 (Security Condition) バイトにより構成される。

DF に対するアクセス制御としてはファイルの消去、有効化/無効化、また該当 DF の下位 DF 作成権、EF 作成権、消去権がある。同様に EF に対してはファイルの消去、有効化/無効化、書き込み・更新・読み込み権がある。アクセス制御はファイルに対するものだけでなく、データオブジェクトについても存在する。データオブジェクトとは、データ内容を示すタグ (Tag) とデータ値の長さ (Length)、データ値 (Value) という構造 (TLV 構造) からなるデータであり、ファイル内容がこれらデータオブジェクトより構成されているものもある。

セキュリティ状態は、ファイルに対するアクセス制御とは別に規定されるもので、ファイルにアクセスするときに必要な状態を示すものである。セキュリティ状態として、カードコマンド自体が暗号化・認証が必要なセキュアメッセージングの利用、IC カードを利用する端末 (PC など) の認証 (外部認証)、カード保有者の認証 (ユーザ認証) があり、さらに詳細にどのような認証情報や暗号化鍵が使われるかなどを示したセキュリティ環境の指定がある。セキュリティ環境に関しては 2.3.3 節で解説する。

### 2.3.3. セキュリティ環境

7816-4 で定義されているセキュアメッセージングの実行や、7816-8 で定義されてい

るセキュリティオペレーション（PERFORM SECURITY OPERATION コマンド）の実行では、暗号化や署名などの操作が行われる。それらを実行するための情報を保持する機構がセキュリティ環境（Security Environment：SE）である。

セキュリティ環境自身は GET DATA コマンドで取得可能である。そしてセキュリティ環境の設定は、MANAGE SECURITY ENVIRONMENT コマンドにより行われる。これらカードコマンドは 7816-4 で定義されているものである。本報告書では 2.4 節でこれらカードコマンドを解説している。

セキュリティ環境は、CRT（Control Reference Template）により表現される。CRT では以下の事柄が記述される。

- 暗号アルゴリズム
- オペレーションモード
- プロトコル
- 手続き
- 鍵
- その他の情報

さらに、CRT は用途ごとに複数の種類に分けられる。

- 認証用 CRT（CRT valid for authentication：AT）
- 鍵共有用 CRT（CRT valid for key agreement：KAT）
- ハッシュ値用 CRT（CRT valid for hash code：HT）
- 暗号チェックサム用 CRT（CRT valid for cryptographic checksum：CCT）
- 電子署名用 CRT（CRT valid for digital signature：DST）
- 機密保持用 CRT（CRT valid for confidentiality）

これら CRT の内部に、実際の鍵などを保管しているファイルへのパスが書かれていたり、直接鍵データが書かれていたりする。セキュリティ環境を利用する場合の概念図として、IC カードのセキュリティ環境を設定する図を図 17 に示す。

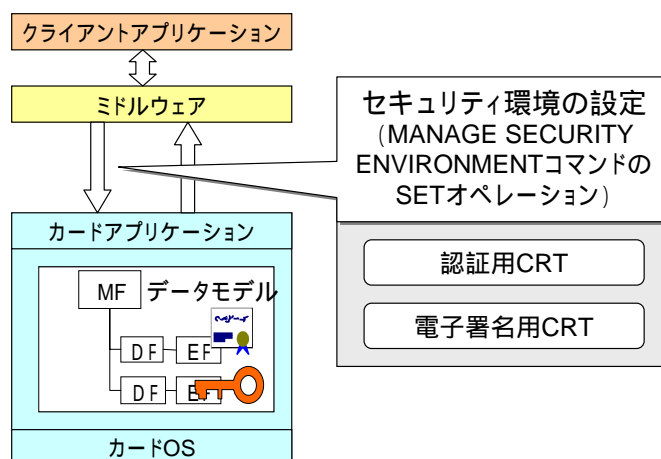


図 17 セキュリティ環境の設定

カード内のデータ構造を定義する 7816-15 では、セキュリティ環境については CIAInfo 内に SecurityEnvironmentInfo という要素を持つ。ここを設定することで、カードに事前設定されるセキュリティ環境情報を伝えることができる。カード内のデータ構造に関しては 2.5 節で解説する。

## 2.4. カードエッジ・インターフェース

本節では、カードエッジ・インターフェースとして、ISO7816 群で定義されているカードコマンドの解説と、汎用カードインターフェースのカードコマンドとして 24727-2 で利用されている 7816 群内のカードコマンドを解説する。

表 8 は ISO7816-4、8 で規定されているカードコマンドの一覧である。

表 8 ISO/IEC 7816-4,8 で規定されているコマンド一覧

コマンド種別、コマンド名	参照標準	コマンドの概要
選択コマンド		
SELECT	ISO/IEC 7816-4	ファイルなどの論理チャンネルを開く
MANAGE CHANNEL		論理チャンネルの管理
データ操作コマンド		
READ BINARY	ISO/IEC	データユニットの読み出し

	WRITE BINARY	7816-4	データユニットの書き出し
	UPDATE BINARY		データユニットの更新
	SEARCH BINARY		データユニットの検索
	ERASE BINARY		データユニットの消去
	READ RECORD		レコードの読み出し
	WRITE RECORD		レコードの書き出し
	UPDATE RECORD		レコードの更新
	APPEND RECORD		レコードの追加
	SEARCH RECORD		レコードの検索
	ERASE RECORD		レコードの消去
	GET DATA		データオブジェクトの取得
	PUT DATA		データオブジェクトの配置
認証関連コマンド			
	INTERNAL AUTHENTICATE	ISO/IEC 7816-4	IC カードまたはカード保有者の 認証
	GET CHALLENGE		IC カードからチャレンジデータ を取得
	EXTERNAL AUTHENTICATE		外部端末の認証
	GENERAL AUTHENTICATE		汎用認証コマンド
	VERIFY		PIN やバイオメトリック情報の確 認
	CHANGE REFERENCE DATA		参照しているデータの変更
	ENABLE VERIFICATION REQUIREMENT		VERIFY 要求を有効化する
	DISABLE VERIFICATION REQUIREMENT		VERIFY 要求を無効化する
	RESET RETRY COUNTER		PIN の再試行カウンタをリセット する
伝送処理関数			
	GET RESPONSE	ISO/IEC	応答 APDU の送信

	ENVELOPE	7816-4	コマンド APDU を別途コマンドデータ内に含み送信する
MANAGE SECURITY ENVIRONMENT(セキュリティ環境の管理)			
	SET	ISO/IEC 7816 - 4	セキュリティ環境のコンポーネントの設定
	STORE		セキュリティ環境の保存
	RESTORE		セキュリティ環境の置き換え
	ERASE		セキュリティ環境の消去
PERFORM SECURITY OPERATION (セキュリティオペレーションの実行)			
	COMPUTE CRYPTOGRAPHIC CHECKSUM	ISO/IEC 7816-8	暗号論的チェックサム (Cryptographic Checksum)の 計算
	COMPUTE DIGITAL SIGNATURE		電子署名の計算
	HASH		ハッシュ値の計算
	VERIFY CRYPTOGRAPHIC CHECKSUM		暗号論的チェックサムの検証
	VERIFY DIGITAL SIGNATURE		電子署名の検証
	VERIFY CERTIFICATE		証明書の検証
	ENCIPHER		暗号化
	DECIPHER		復号化
	GENERATE ASYMMETRIC KEY PAIR		非対称鍵の生成

すべてのカードコマンドとその応答のフォーマットは共通になっている(図 18)。各フォーマットにおけるデータの解説を表 9 に示す。

コマンドフォーマット

CLA	INS	P1	P2	Lc	コマンドデータ	Le
-----	-----	----	----	----	---------	----

応答フォーマット

応答データ	SW1	SW2
-------	-----	-----

図 18 カードコマンドと応答のフォーマット

表 9 各フォーマットにおけるデータ

コマンド	CLA	クラスバイト。各ビットに意味を持つ。最上位ビットがセットされている場合、7816 標準のコマンドではないことを示す。
	INS	処理を行うコマンドを示す。
	P1	コマンドに付随するパラメータを示す。
	P2	
	Lc	後に続くコマンドデータの長さを示す。
	コマンドデータ	
	Le	応答データで予想される最大のデータ長。
応答	応答データ	
	SW1	コマンド実行結果のステータス（成功・エラー）を示す。
	SW2	

本節では、7816 で規定されているカードコマンド群の内、IC・ID カードの相互運用可能性で特に重要と思われるカードコマンドについて詳細に解説を行う。また、2.4.7 節においては、ISO/IEC 24727-2 で利用されているカードコマンドの解説を行う。

#### 2.4.1. SELECT コマンド

SELECT コマンドは ISO/IEC 7816-4 で規定されているカードコマンドである。このコマンドにより、その後の処理を行う対象となるファイルを指定する。ファイルの指定は、ファイル識別子や DF 名、パスの 3 種類が指定できる。また選択するファイルの種類や、応答で得られるファイル管理情報（File Control Information：FCI）の種類を指定する。ファイル管理情報には 2.3.2 節で解説したセキュリティ属性も含まれている。

3.4 節で解説する PIV はファイルを持つ構造を持っておらず、アプリケーション識別子 (AID、2.3.1 節参照) を指定するが、SELECT コマンドはパラメータの指定 (P1 を 04h に指定) により AID を利用してデータ選択をすることが可能である。

#### 2.4.2. データ操作コマンド

データを扱うカードコマンドに関して、ISO/IEC 7816-4 ではデータ種類に応じたカードコマンドを定義している。そのデータ種類として、データユニット、レコード、データオブジェクトがある。それぞれのデータの種類の種類は、EF がどういったデータを扱うかで決定される。

EF が単一の連続データ列として見える透過構造の場合、データユニットとして利用される。データユニットに対するカードコマンドとして、READ BINARY、WRITE BINARY、UPDATE BINARY、ERASE BINARY がある。

一方ある長さのデータが番号付けや識別子を付与されているものをレコードといい、レコードを利用する EF に対するカードコマンドとして、READ RECORD、WRITE RECORD、UPDATE RECORD、APPEND RECORD、SEARCH RECORD がある。

また、データが TLV 構造になっているものをデータオブジェクトと呼ぶ。データオブジェクトに対するカードコマンドとして、GET DATA と PUT DATA がある。

EF がレコードで構成されている場合や TLV 構造である場合でも、それを単一の連続データ列として READ BINARY、WRITE BINARY、UPDATE BINARY、ERASE BINARY コマンドを利用することも可能である。

#### 2.4.3. 認証用コマンド

IC・ID カードにおいてもっとも重要な認証の対象は、IC・ID カードを保持する IC カード保有者であるが、IC カードが認証に用いられる場合、認証の対象となるものが複数あることに注意されたい。1 つは、IC カード自身である。これは IC カードの保有者を認証するのではなく、外部端末が IC カードを認証するものである。そしてもう 1 つの対象が、IC カードを利用する外部端末である。外部端末への認証 (外部認証) は、IC カードにより行われ、たとえば重要なデータの更新に外部認証を行った端末でなければ行われなくすることなどがある。

表 10 IC カードにおける認証の種類

認証を行うエンティティ	認証の対象となるエンティティ
外部端末	IC カード
IC カード	IC カード保有者
IC カード	外部端末

ISO/IEC 7816-4 では、それらの認証を行うカードコマンドとして INTERNAL AUTHENTICATE、EXTERNAL AUTHENTICATE、GENERAL AUTHENTICATE、そして VERIFY の 4 つのカードコマンドを規定している。

INTERNAL AUTHENTICATE は、外部端末が IC カードを認証するときに用いるカードコマンドである。コマンドデータにチャレンジデータを入れ、チャレンジデータを受け取った IC カードは、IC カードのみが所持する情報（プライベート鍵など）や IC カードと外部端末が共有する秘密情報（共通鍵）を用いてレスポンスデータを計算し、応答データとして外部端末に返す。外部端末はそれら応答データを検証することで IC カードの認証を行う。

EXTERNAL AUTHENTICATE は、IC カードが外部端末を認証（外部認証）するときに用いるカードコマンドである。実際にはこのカードコマンドだけでは IC カードが外部端末を認証することではなく、このカードコマンドに先立ち GET CHALLENGE コマンドを送信することで、IC カードよりチャレンジが送られてくる。そして外部端末でそのチャレンジに対するレスポンスを計算し、EXTERNAL AUTHENTICATE コマンドのコマンドデータにおいてそのレスポンスを IC カードに送信する。IC カードはそのレスポンスを確認することで、外部端末の認証を行う。カード発行や管理を行う際に使われることがある。

GENERAL AUTHENTICATE は INTERNAL AUTHENTICATE と EXTERNAL AUTHENTICATE コマンドを改良したものであり、双方の機能を持つカードコマンドである。さらに、GENERAL AUTHENTICATE コマンドでは、INTERNAL AUTHENTICATE と EXTERNAL AUTHENTICATE の 2 つのカードコマンドの組み合わせでは実現できなかった、より複雑な認証プロセスを実現できるようになっている。3.4 節で解説する PIV では GENERAL AUTHENTICATE コマンドを利用することで電子署名や暗号化を行っている。

#### 2.4.4. VERIFY コマンド

VERIFY コマンドは、IC カード外から入力された PIN や、指紋などのバイオメトリック情報を検証するためのカードコマンドである。証明書や電子署名の検証などは VERIFY コマンドでは行えず、2.4.6 節で解説する PERFORM SECURITY OPERATION コマンドで行う。たとえば電子署名を行う場合などは、プライベート鍵へのアクセスを行う場合に PIN による認証が必要とされるために、まず VERIFY コマンドでプライベート鍵オブジェクトに対するアクセスを可能にした後に署名操作を行っている。

VERIFY コマンドは ISO/IEC 7816-4 で規定されている。

#### 2.4.5. MANAGE SECURITY ENVIRONMENT コマンド

カードコマンド自身を暗号化するなどのセキュアメッセージングや、電子署名や認証などのセキュリティに関連したカードコマンド群の利用時には、その実行のために利用される暗号化の鍵や認証情報などを示さなければならない。そういった情報は 2.3.3 節で解説するセキュリティ環境で提供される。MANAGE SECURITY ENVIRONMENT コマンドは、セキュリティ環境を管理するカードコマンドであり、表 11 に示す機能を持つ。

表 11 MANAGE SECURITY ENVIRONMENT コマンドの機能

機能	概要
SET	現在のセキュリティ環境内の 1 つのコンポーネントを設定・交換する。
STORE	現在のセキュリティ環境を、コマンドパラメータ P2 で指定されたセキュリティ環境 ID (SEID) で保存する。
RESTORE	現在のセキュリティ環境を、カード内に保管されているセキュリティ環境と交換する。セキュリティ環境の指定はコマンドパラメータ P2 にセキュリティ環境 ID を指定することで実現する。
ERASE	カード内に保管されているセキュリティ環境を消去する。セキュリティ環境の指定はコマンドパラメータ P2 にセキュリティ環境 ID を指定することで実現する。

それぞれの機能を実行する場合は、コマンドパラメータ P1 で設定する。STORE、RESTORE、ERASE を実行する場合は、コマンドパラメータ P2 において対象となるセキュリティ環境の ID (SEID) を指定する。SET を実行する場合は、コマンドパラメータ P2 において、どの CRT を設定するかを指定する。

たとえば電子署名を行う場合、セキュリティ環境がすでに設定してあれば GET DATA コマンドでセキュリティ環境を呼び出し、そこに記述されている情報を利用してプライベート鍵へのアクセスを行い、署名操作を行うことができる。しかし、セキュリティ環境が設定されていない場合や、設定されていても適切ではない（電子署名用の CRT が設定されていない）場合は、SET オペレーションあるいは RESTORE オペレーションにより電子署名用の CRT を設定することが必要になる。次節で解説する PERFORM SECURITY OPERATION コマンドでは、電子署名の計算や検証などプライベート鍵・公開鍵を使う操作の中で、複数の鍵から特定の鍵を選択する機能を持っていない。そのため PERFORM SECURITY OPERATION に先立って、MANAGE SECURITY ENVIRONMENT コマンドにより適切な鍵を設定することが必要になる。

MANAGE SECURITY ENVIRONMENT コマンドは ISO/IEC 7816-4 で規定されている。

#### 2.4.6. PERFORM SECURITY OPERATION コマンド

ISO/IEC 7816-8 で規定されている PERFORM SECURITY OPERATION コマンドは、さまざまなセキュリティオペレーションを行う。このカードコマンドで行われるセキュリティオペレーションを表 12 に示す。

表 12 PERFORM SECURITY OPERATION コマンドのオペレーション

オペレーションの種類	概要
COMPUTE CRYPTOGRAPHIC CHECKSUM	暗号論的チェックサム (Cryptographic Checksum) の計算
COMPUTE DIGITAL SIGNATURE	電子署名の計算
HASH	ハッシュ値の計算
VERIFY CRYPTOGRAPHIC CHECKSUM	暗号論的チェックサムの検証

VERIFY DIGITAL SIGNATURE	電子署名の検証
VERIFY CERTIFICATE OPERATION	証明書を検証
ENCIPHER	暗号化
DECIPHER	復号

それぞれのオペレーションは、コマンドパラメータ P1 と P2 により指定される。またそれぞれのオペレーションを実行するにあたっては、いくつかのカードコマンドが先立って完了していることが求められる。MANAGE SECURITY ENVIRONMENT などがそれにあたる。それぞれのオペレーションと P1、P2 の値を表 13 に示す。

表 13 オペレーションごとの P1/P2 値

オペレーション	P1	P2
COMPUTE CRYPTOGRAPHIC CHECKSUM	8E	80
COMPUTE DIGITAL SIGNATURE	9E	9A,AC,BC
HASH	90	80,A0
VERIFY CRYPTOGRAPHIC CHECKSUM	00	A2
VERIFY DIGITAL SIGNATURE	00	A8
VERIFY CERTIFICATE	00	92,AE,BE
ENCIPHER	82,84,86	80
DECIPHER	80	82,84,86

表 13 では同じオペレーションでも複数の P1、P2 値があるものがあるが、これは基本的に入力するデータの形式の差を包含するものとなっている。たとえば電子署名の計算を行う COMPUTE DIGITAL SIGNATURE オペレーションでは P2 に 3 つの値を利用している。これは大きく分けて、入力するコマンドデータが電子署名されるデータそのものか、または TLV 形式をもつデータオブジェクトとなっているかを示すものとなっていることなどがある。

#### 2.4.7. ISO/IEC 24727-2 で利用されるカードコマンド群

ISO/IEC 24727-2 は、2.1.1 節で解説した 24727-1 のアーキテクチャにおける汎用カードアクセスレイヤーに関する詳細仕様を規定したものであり、汎用カードインターフ

エースで利用されるカードコマンドや、IC カード内に存在する機能表示（Capability Description）が規定されている。機能表示は、カードアプリケーションの機能表示を行う ACD（Application Capability Description）とカードの機能表示を行う CCD（Card Capability Description）の 2 種類ある。CCD は IC カード内に 1 つのみ存在し、ACD は各カードアプリケーションに存在する。

ISO/IEC 24727-2 の汎用カードインターフェースで規定されているカードコマンドは、ISO/IEC 7816 で規定されているカードコマンド群の部分集合である。これらのカードコマンドの選択が IC・ID カードサービスの相互運用を実現するために選ばれていることに注目したい。24727-2 で規定されているカードコマンドの一覧を表 14 に示す。24727-2 で規定されているコマンドは、以前の草案ではすべて必須コマンドとされていたが、2006 年 11 月時点の草案ではそれらは必須ではなく「利用されるべき」となっている。

表 14 ISO/IEC 24727-2 で規定されているカードコマンド

コマンド種別、コマンド名		参照標準
選択コマンド		
	SELECT	ISO/IEC 7816-4
データ操作コマンド		
	READ BINARY	ISO/IEC 7816-4
	UPDATE BINARY	
	GET DATA	
	PUT DATA	
認証関連コマンド		
	VERIFY	ISO/IEC 7816-4
	CHANGE REFERENCE DATA	
MANAGE SECURITY ENVIRONMENT		
	SET	ISO/IEC 7816-4
	RESTORE	
PERFORM SECURITY OPERATION		

	COMPUTE DIGITAL SIGNATURE	ISO/IEC 7816-8
	HASH	
	VERIFY DIGITAL SIGNATURE	
	VERIFY CERTIFICATE	
	ENCIPHER	
	DECIPHER	
GENERATE ASYMMETRIC KEY PAIR		
	CREATE FILE	ISO/IEC 7816-9
	DELETE FILE	
	ACTIVATE FILE	
	DEACTIVATE FILE	
	RESET	

## 2.5. データモデル

暗号化や認証、電子署名を実現するための情報を保持するカードアプリケーションの標準化として、ISO/IEC 7816-15 がある。7816-15 はもともと RSA セキュリティ社が規定した PKCS # 15<sup>[13]</sup>を基に標準化されたものであるが、PKCS#15 とは多少異なる部分がある。表 15 はデータモデルに関する標準や本報告書で紹介する事例などの策定期間を時系列に示したものである。本小節では ISO/IEC 7816-15 を中心に解説し、その後 PKCS#15 との差異を示す。

表 15 ISO/IEC 7816-15 に関連する仕様・標準

策定された時期		策定された仕様
1999 年	4 月	PKCS#15v1.0
	10 月	FINEID 仕様 v1.0 (PKCS#15 対応)
	12 月	PKCS#15v1.0 改訂版 1
2000 年	6 月	PKCS#15v1.1
2004 年	1 月	ISO/IEC 7816-15:2004
2006 年	2 月	JIS X 6320-15:2006

### 2.5.1. ISO/IEC 7816-15

ISO/IEC 7816-15 は、カードアプリケーションを規定した規格である。そのアプリケーションは暗号や認証、電子署名を行うために必要な情報を含む機構を提供する。ここで気をつけなければならないのは、IC カード上にある ISO/IEC 7816-15 に従って表現された情報は、IC カードの動作や特性を情報として表現しているのみであり、7816-15 で表現された情報を書き換えたところで IC カードの動作や特性は変更されない。

ISO/IEC 7816-15 ではそういった情報を暗号情報 (Cryptographic Information) と呼んでいる。暗号情報はデータオブジェクトであり、暗号情報オブジェクト (CIO) はその情報内容によりさらに細分化されたオブジェクトとして規定される。CIO の論理構造を図 19 に示す。

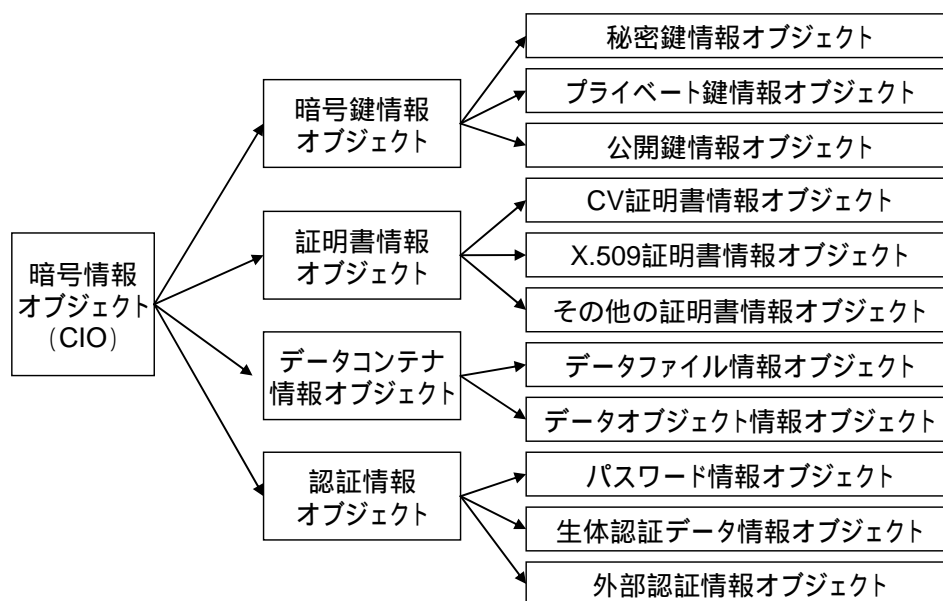


図 19 暗号情報オブジェクトの論理構造

実際に IC カード内のデータとしてこれらの情報が保持される場合のファイル構造は図 20 のようになる。ここで MF はカード内のファイルシステムの最上位にあるマスターファイルを示す。また EF は基礎ファイルを示し、DF は専用ファイルを示す。各ファイルの後に続く文字はオブジェクトの種類を示している。たとえば EF.AOD は認証オブジェクトディレクトリを示し、EF.PrKD はプライベート鍵、EF.PuKD は公開鍵、

EF.CD は証明書のそれぞれの情報オブジェクトとなっている。また DF.CIA の CIA は暗号情報アプリケーション（Cryptographic Information Application）を示すものであり、7816-15 が規定しているカードアプリケーションそのものである。

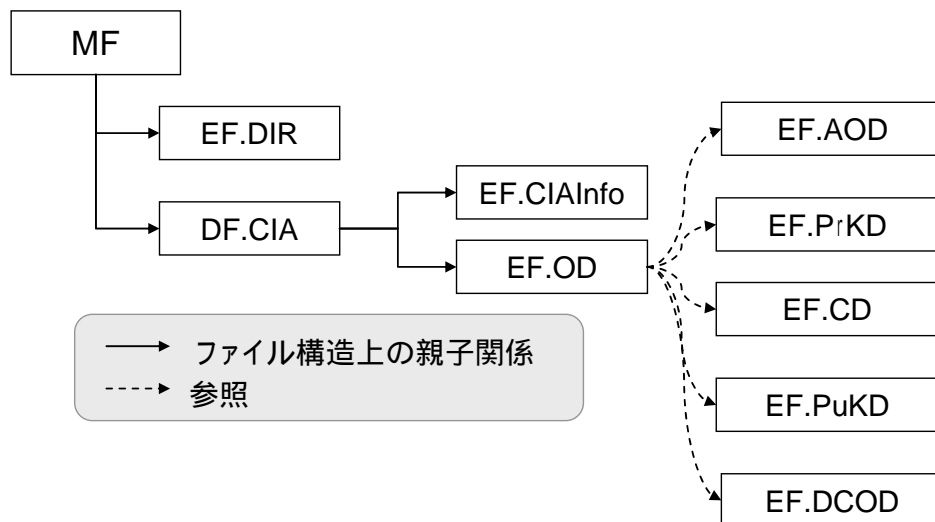


図 20 暗号情報オブジェクトのファイル構造

本節ではいくつかのファイルや情報オブジェクトについて解説を行う。表 16 に解説するオブジェクトの概要を示す。また、それぞれのオブジェクトはいくつかの属性をもつが、IC・ID カードの相互運用可能性に重要と思われるものに関していくつか解説を行う。

表 16 各オブジェクトの概要

オブジェクト	説明
EF.DIR カードアプリケーション情報	ISO/IEC 7816-15 で規定されているものではなく、7816-4 で規定されているものであり、カードアプリケーションの選択に利用される。
DF.CIA 暗号情報アプリケーションディレクトリ	ISO/IEC 7816-15 で規定するアプリケーションの DF であり、ここを基点としてさまざまな暗号情報オブジェクトへとアクセスを行う。
EF.CIAInfo 暗号情報アプリケーション情報	CIA のバージョンやシリアル番号など CIA 自身を示すものに加え、カード自体の情報を含む。

EF.OD オブジェクトディレク トリ	CIA がどの種類の暗号情報オブジェクトを含むかを示し、さらにそれぞれの暗号情報オブジェクトへの参照を示す。
EF.AOD 認証情報オブジェクト	IC・ID カードがカード保有者を認証するための PIN などが格納される。複数の PIN を持つ場合もある。
EF.PrKD プライベート鍵情報オ ブジェクト	公開鍵暗号のプライベート鍵が格納され、署名、復号に利用される（署名のみに制限される場合もある）。
EF.PuKD 公開鍵情報オブジェク ト	公開鍵ペアにおける公開鍵の情報とその鍵自身を示すファイルである。通常はあまり利用されない。
EF.CD X509 証明書情報オブジ ェクト	X.509 公開鍵証明書が格納される。カード保有者の証明書や、同じくカード保有者の信頼点の証明書（自己署名証明書）等が格納される。

## EF.DIR

EF.DIR は ISO/IEC 7816-15 で規定されているものではなく、7816-4 で規定されているものであり、複数のカードアプリケーションを持つ場合にその AID を保持することでカードアプリケーションの選択を可能にしている。MF 下に EF.DIR が存在する場合、ISO/IEC 7816-15 で規定するアプリケーション、つまり CIA の AID も記述される。CIA の AID は、最初の 5 バイトが「E8 28 BD 08 0F」と規定されている。一方で、ISO/IEC 7816-15 内には備考として CIA の AID に関して「検討の経緯から、AID:”A0 00 00 00 63 50 4B 43 53 2D 31 35”を使用して DF.CIA を選択してもよい。」と記述してある。この AID は PKCS#15 を示すものであり、ISO/IEC 7816-15 は PKCS#15 を基に策定された規格であることから、PKCS#15 アプリケーションとして読み込むミドルウェアやシステムに対応するためのものと考えられる。なお、後半の 7 バイトが ASCII コードで「PKCS-15」を示すものとなっている。

AID の指定とともに、次に説明する DF.CIA へのパスが記述されている。

## DF.CIA

ISO/IEC 7816-15 で規定するアプリケーションの DF であり、ここを基点としてさまざまな暗号情報オブジェクトへとアクセスを行う。DF.CIA の下には必ず EF.CIAInfo

と EF.OD が存在しなければならない。それら 2 つのファイル識別子は指定されており、EF.CIAInfo では 5032、EF.OD では 5031 となっている。

### **EF.CIAInfo**

EF.CIAInfo は CIA のバージョンやシリアル番号など CIA 自身を示すものに加え、カード自体の情報を含む。

バージョンは v1 (データ上は 0)、v2 (データ上は 1) の 2 種類が指定可能であるが、ISO/7816-15 では v2 を利用するように規定されている。v1 は PKCS#15 で使用しているためである。

カード自体の情報として必ず含まなければならないものとして、カードフラグがある。これはカード自身が読み取り専用 (ReadOnly) であることや、利用者の認証を要求する (authRequired) ことなどが示される。

EF.CIAInfo ではさらにセキュリティ環境 (SE) の情報を含むことが可能である。これはカードに事前設定されているセキュリティ環境を示すものである。CIAInfo で示されるセキュリティ環境の情報は、SEID とカード保有者やカード発行者などの所有者情報、さらにセキュリティ環境が適用可能なカードアプリケーションを示す AID から構成される。

### **EF.OD**

EF.OD は、CIA がどの種類の暗号情報オブジェクトを含むかを示し、さらにそれぞれの暗号情報オブジェクトへの参照を示す。EF.OD で示される暗号情報オブジェクトは、通常オブジェクトが存在するファイルのパスが指定されるが、EF.OD が直接所持してもよい仕様となっている。

### **EF.AOD**

EF.AOD は、認証方法に関する情報を提供するファイルである。ここで言う認証には、オブジェクトへアクセスする際に必要となる認証と、IC カードが外部端末を認証するための外部認証の 2 つがある。オブジェクトへアクセスする際の認証方法用として、パスワードと生体認証の 2 つのオブジェクトが規定されている。

### **EF.PrKD**

EF.PrKD は公開鍵ペアにおけるプライベート鍵の情報とその鍵自身を示すファイルである。どの暗号アルゴリズムの鍵であるかが指定され、さらにそれぞれのアルゴリズム

ムに従ったプライベート鍵情報が指定される。代表的な RSA 暗号の場合、指定される情報はプライベート鍵を含むファイルへのパス、鍵長などが含まれる。ISO/IEC 7816-15 で選択できるプライベート鍵のアルゴリズムとしては RSA のほかに、楕円曲線、Diffie-Hellman、DSA、KEA が規定されているほか、他のアルゴリズムでも利用できるように汎用オブジェクトも用意されている。

### **EF.PuKD**

EF.PuKD は公開鍵ペアにおける公開鍵の情報とその鍵自身を示すファイルである。どの暗号アルゴリズムの鍵であるかが指定され、さらにそれぞれのアルゴリズムに従った公開鍵情報が指定される。代表的な RSA 暗号の場合、指定される情報は RSA 公開鍵への参照または公開鍵データ、鍵長などが含まれる。ISO/IEC 7816-15 で選択できる公開鍵のアルゴリズムとしては、PrKD で選択できるアルゴリズムと同様となっている。

### **EF.CD**

EF.CD は証明書の情報を示すファイルである。まず証明書の形式が指定され、さらにそれぞれの形式に従った情報が指定される。代表的な X.509 証明書の場合、X.509 証明書自身が存在しているファイルへのパスが必須情報となっている他、証明書の所有者名 (Subject) 発行者名 (Issuer) シリアル番号がオプションとして指定可能である。ISO/IEC 7816-15 で選択できる証明書の形式として X.509 証明書のほか、X.509 属性証明書、SPKI 証明書、PGP 証明書、WTLS 証明書、ANSI X9.68 ドメイン証明書、カード検証可能証明書が規定されているほか、他の証明書形式でも利用できるように汎用オブジェクトも用意されている。

さまざまな暗号情報オブジェクトは、多種多様な属性を持つが、IC・ID カードの相互運用可能性において特に重要だと思われるいくつかの属性について下に解説する。

### **flags**

flags はすべてのオブジェクトに共通する属性であり、3 つのビットによりそのオブジェクトの特性を示す。それぞれのビット名称と意味を表 17 に示す。flags の利用は必須ではない (OPTIONAL)。

表 17 flags 属性に含まれる各ビットの意味

ビット名称	意味
private	パスワードなどの認証の後にだけオブジェクトがアクセス可能
modifiable	オブジェクトの値を変更することができる
internal	ISO/IEC 7816-15 で規定されているが、「歴史的理由で存在しているが、使用しないほうがよい」とされている。

### authId

authId は EF.AOD 下にある認証オブジェクトのそれぞれに付与されており、一意なものでなければならない。

flags において private のビットが設定されているオブジェクトの場合、プライベートオブジェクトへのアクセスには認証が必要であり、authId で指定された認証オブジェクトを利用して認証を行う。

たとえば、プライベート鍵オブジェクトにおいて flags に private が指定され authId が 1 と指定されていた場合、まず authId が 1 である認証オブジェクトにより認証を行い、その後にプライベート鍵オブジェクトが利用される。

### userConsent

userConsent はすべてのオブジェクトに共通する属性であり、オブジェクトへのアクセスに対して利用者の明示的な同意要求なしで出来る回数を指定できる。userConsent が 1 に指定してある場合、そのオブジェクトに対するアクセスを行うたびに利用者の認証を必要とされる。電子署名を行うためのプライベート鍵の利用において、userConsent 属性が設定されることがある。その際、flags に private が指定され、authId が指定されており、さらにプライベート鍵オブジェクトの userConsent の値に 1 が指定される。そうすることで、プライベート鍵へのアクセスを行うたびに、authId で指定された認証オブジェクトによる認証 (PIN など) により認証を受けなければならなくなる。

### usage

usage は暗号鍵情報オブジェクトに共通する属性であり、暗号鍵の使用目的を示すビット列となっている。それぞれのビットが設定されることでその使用目的を示す。ビットと仕様目的の対応を表 18 に示す。

表 18 usage 属性の各ビット名称

ビット	名称
0	encipher
1	decipher
2	sign
3	signRecover
4	keyEncipher
5	keyDecipher
6	verify
7	verifyRecover
8	derive
9	nonRepudiation

### authority

authority は証明書情報オブジェクトに共通する属性であり、その証明書が認証局 (CA) などの機関 (Authority) 向けであることを示す BOOLEAN 型のデータである。たとえば、IC カード内にある信頼点となるルート CA の自己署名証明書で利用される。

### 2.5.2. ISO/IEC 7816-15 と PKCS#15 の差異

ISO/IEC 7816-15 は RSA セキュリティ社が規定した仕様である PKCS#15 の Version1.1 を基に策定されているが、双方にはいくつかの違いがある。本小節ではそれらの差異について解説する。

データの構造として、双方が持つオブジェクトの種類に大差はない。唯一 PKCS#15 には UnusedSpace という EF を持つことが書かれている。UnusedSpace は、既に作成されている EF 内の未使用スペースを指し示すポインタが格納された EF であり、オブジェクトの削除や追加時にオブジェクトのポインタを示すファイルとともに書き換えられる。PKCS#15 には、オブジェクトの追加時や削除時での UnusedSpace の利用方法も示されている一方で、ISO/IEC 7816-15 では UnusedSpace は書かれていなく、未使用スペースに関しては参照先 EF の未使用スペースを「FF」で埋めることが示されているのみである。

また usage 属性のそれぞれのビット名称も異なっている。表 19 にその一覧を示す。

表 19 usage ビット名称の差異

ビット	PKCS#15	ISO/IEC7816-15
0	encrypt	encipher
1	decrypt	decipher
2	sign	sign
3	signRecover	signRecover
4	wrap	keyEncipher
5	unwrap	keyDecipher
6	verify	verify
7	verifyRecover	verifyRecover
8	derive	derive
9	nonRepudiation	nonRepudiation

データを表現するときに、ISO/IEC 7816-15 の中には「歴史的理由により \* \* \* は使用されない」や「歴史的理由により残されているが、利用されるべきではない」などの記述が散見されるが、これは PKCS#15 に存在するものの ISO/IEC 7816-15 では規定しない、あるいは利用しないものという差異であり、ISO/IEC 7816-15 ではこれらを記述することで PKCS#15 との相互運用を図っている可能性がある。

## 2.6. カードサービスの API

2.3 節から 2.5 節で解説した事柄は主にカード内部に直接アクセスするためのものであったが、クライアントアプリケーションが IC カードにアクセスする際には意識しない部分でもある。クライアントアプリケーションはより高度に抽象化された概念を用いてプログラミングされるのが一般的である。たとえばカードに対してカード保有者 PIN のペリファイ、データの暗号化といった操作を行う一連の操作を、ミドルウェアによって「署名操作」として提供された API を用いて実行する。カード内部に対するアクセスやデータモデルの理解はミドルウェアが行い、クライアントアプリケーションはミドルウェアが提供する API を利用して設計・構築される。

ミドルウェアが提供する API に関する標準は、ISO/IEC 24727-3 で標準化作業が進んでいる。また、事実上の標準として、暗号化や署名を行うトークンへの API を規定した

PKCS#11 が広く使われている。本小節ではこれらの解説と、違いについて考察する。

### 2.6.1. ISO/IEC 24727-3

ISO/IEC 24727-3 はクライアントアプリケーションが IC カードサービスを利用するときのインターフェースを規定したものである。具体的には IC カードサービスを扱うための API が規定されている。多くの関数が規定されているが、大きく分けて以下の様に分類される。

- カードへの接続  
初期化・切断や、カードアプリケーションへの接続などの関数がある。
  
- カードアプリケーションの利用  
カードアプリケーション一覧の取得、カードアプリケーションの作成・削除、アプリケーションの実行などの関数がある。
  
- データセットの処理  
データセット一覧の取得、データセットの作成・選択・削除などの関数がある。
  
- 暗号サービス  
チャレンジの取得、署名検証、署名、暗号化・復号化の関数がある。
  
- 認証対象の処理  
認証の対象となる差分アイデンティティ ( Differential-Identity ) の一覧取得・作成・削除などに加え、認証を行う関数が含まれる。
  
- 認可サービス  
アクセス制御リストの一覧取得と、アクセスルールの更新を行う関数が含まれる。

### 2.6.2. PKCS#11

PKCS#11 は、PKCS#15 と同様に RSA セキュリティ社により規定された仕様であり、トークンに対して証明書の処理や公開鍵暗号の鍵ペアを利用した演算を行うための API

が規定されたものである。

PKCS#11に関する概要は、「セキュリティ API に関する技術調査」中の「Part4. IC カードなどのハードウェアトークン API」を参照されたい。

### 2.6.3. 24727-3 と PKCS#11 の違い

ISO/IEC 24727-3 と PKCS#11 ではともに API を規定しているが、その粒度は大きく異なる。たとえば暗号に関連する関数では、24727-3 では暗号化・復号化、署名計算と検証、チャレンジの取得があるが、PKCS#11 ではさらに詳細にメッセージダイジェストに関する計算や乱数生成に関する関数などを含んでおりその機能が詳細にわたっている。ISO/IEC 24727 が規定しているのは IC カードが提供するサービスの相互運用に関連するものであり、PKCS#11 とはその目的が異なっていることが大きな原因であると考えられる。

## 3. IC・IDカードの実装例

### 3.1. IDカードの動向

本節ではIDカードの動向として、フィンランドの国民IDカードプロジェクトである FINEID、ベルギーの国民IDカードプロジェクトである BELPIC、米国の連邦政府職員用IDカードプロジェクトである PIV について解説する。これらは仕様が公開されており、また、2章で述べた標準技術の多くが適用されている。しかし、標準を適用しているとは言え差異は存在する。それらの差異もあわせて解説する。

### 3.2. FINEID

#### 3.2.1. プロジェクト概要

FINEID はフィンランドにおける国民用IDカードのプロジェクトであり、1998年に各省庁を横断するプロジェクトとしてスタートしたもので、ヨーロッパにおいて国民電子IDカードを発行した最初のプロジェクトである。1999年12月に最初のカードがフィンランド首相に発行された。FINEID が提供するのは公共サービスへのアクセス用の国民IDカードであり、オープンで安全でないネットワークにおいて安全性を提供するための基盤作りを大きな目的としている。また、国民登録局 (PRC) DB にある国民データと連結したカードが2000年より導入されている。

#### 3.2.2. 技術概要

FINEID は仕様を公開している。表 20 にその一覧を示す。

FINEID で特徴的なこととして、IDカードにはカード保有者の2種類の鍵ペアと証明書が入っていることである。これらの鍵ペア・証明書はそれぞれ認証・暗号化用と、否認防止の署名用に利用されるものである。

3.2.3 節において、ISO/IEC 7816-4<sup>[8]</sup>、8<sup>[29]</sup>に準拠したカードコマンドを示した「FINEID S1 Electronic ID Application v2.1<sup>[32]</sup>」と、ISO/IEC 7816-15<sup>[14]</sup>に準拠したカード内部のデータフォーマットを示した「FINEID S4-1 FINEID Implementation profile 1 for Finnish Electronic ID Card v2.1A<sup>[33]</sup>」について解説する。

表 20 FINEID 仕様一覧

FINID 仕様名	概要
FINEID S1 Electronic ID Application v2.1	カードコマンドインターフェースと、AID、利用される暗号情報オブジェクトについての概要。
FINEID S2 VRK(PRC) CA-model and certificate contents v2.0	FINEID で利用される証明書の種類とその認証局に関する仕様。
FINEID S4-1 FINEID Implementation profile 1 for Finnish Electronic ID Card v2.1A	鍵や証明書など、FINEID カードで保持される暗号情報の種類とその仕様。さらに ISO/IEC 7816-15 に沿ったオブジェクトの配置とアクセスコンディションの規定。
FINEID S4-2 FINEID Implementation profile 2 for Organizational Usage v2.1A <sup>[35]</sup>	S4-1 に加え、政府組織 (Organizational) で利用する場合の追加仕様。
FINEID S5 Directory Specification v2.1 <sup>[36]</sup>	FINEID のディレクトリ仕様。

FINEID で発行される証明書に関連する信頼モデルを図 21 に示す。

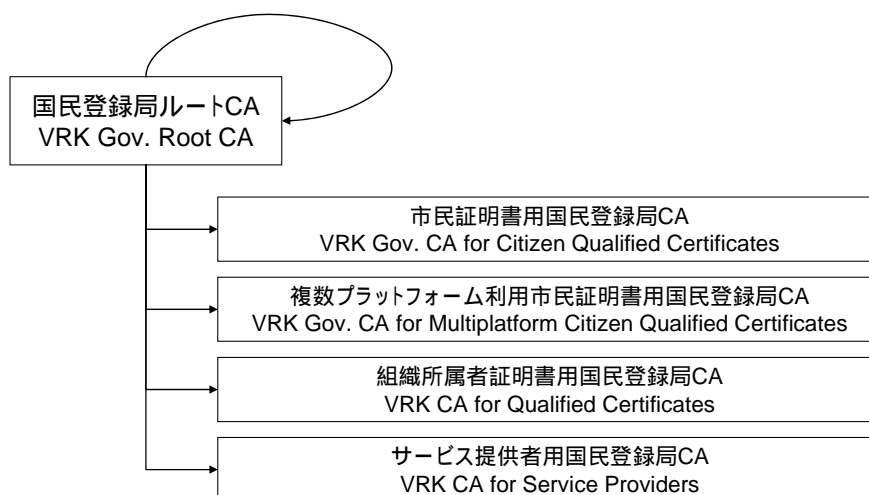


図 21 FINEID の信頼モデル

FINEID で利用される証明書の種類と関連する認証局に関しては「FINEID S2 VRK(PRC) CA-model and certificate contents v2.0」<sup>[34]</sup>に記述されている。それぞれの CA は国民登録局が管理している CA であり、発行する証明書ごとに分割されている。発行される証明書の種類を以下に示す。

- 市民証明書 (Citizen Certificate)
- 組織所属者証明書 (Organization Certificate)
- サーバ証明書 (Server Certificate)
- 電子メール証明書 (E-mail Certificate)

また証明書プロファイルも示されている。CP・CPS については別途公開されている。

個人向けの証明書は市民証明書と組織所属者証明書の 2 種類。電子メール証明書も個人に結び付けられるとも言えるが、commonName の部分がメールアドレスとなっているために厳密には個人向けの証明書とは言えない。組織所属者証明書は、所属する組織 (Organization) が証明書の所有者情報に書かれている。おそらく政府機関に所属する個人に対する発行を想定しているものであろう。

市民証明書と組織所属者証明書はそれぞれの用途の CA より発行される。サーバ証明書と電子メール証明書はサービス提供者用 CA より発行される。

### 3.2.3. カードコマンドとデータ構造、アクセス管理

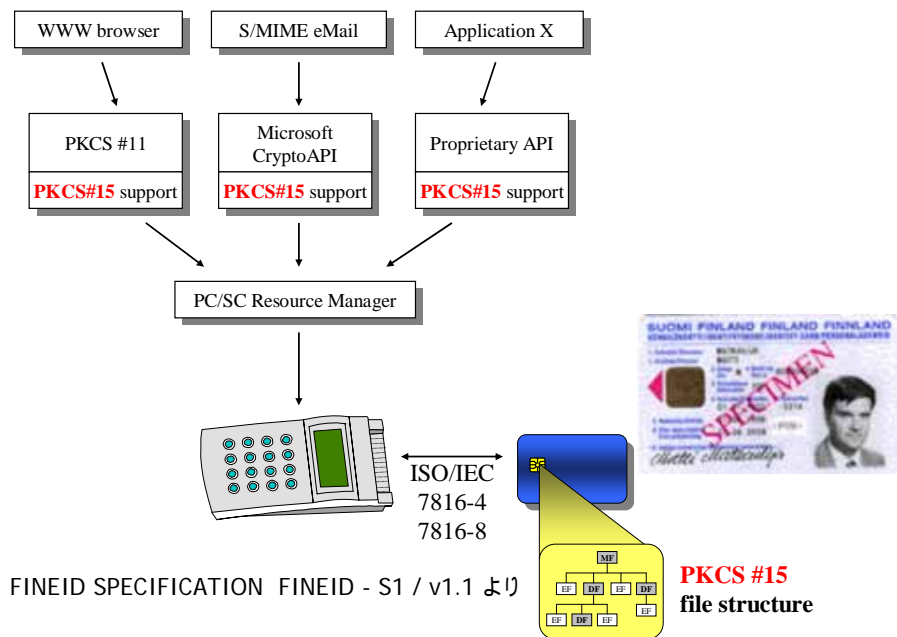


図 22 FINEID カード概要

### カードコマンド

「FINEID S1 Electronic ID Application v2.1」で定義されているカードコマンド群は ISO/IEC 7816-4 と 7816-8 のカードコマンド群より選択されたサブセットとなっている。表 21 に FINEID で定義されているカードコマンド群の一覧を示す。

表 21 FINEID カードコマンド一覧

コマンド種別、コマンド名		参照標準
選択コマンド		
	SELECT	ISO/IEC 7816-4
	SELECT FILE	
データ操作コマンド		
	READ BINARY	ISO/IEC 7816-4
	UPDATE BINARY	
	ERASE BINARY	
	GET DATA	
認証関連コマンド		
	VERIFY	ISO/IEC

	CHANGE REFERENCE DATA	7816-4
	RESET RETRY COUNTER	
伝送処理関数		
	GET RESPONSE	ISO/IEC 7816-4
MANAGE SECURITY ENVIRONMENT		
	SET	ISO/IEC
	RESTORE	7816-4
PERFORM SECURITY OPERATION		
	COMPUTE DIGITAL SIGNATURE	ISO/IEC 7816-4
	HASH	
	DECIPHER	
GENERATE ASYMMETRIC KEY PAIR		

カードコマンド群にはいくつか特徴がある。まず、SELECT と SELECT FILE の双方を持つ点である。FINEID のドキュメントには、SELECT は OpenPlatform の仕様 (バージョン 2.0.1) に沿ったものであり、SELECT FILE は ISO/IEC 7816-4 に沿ったものである旨が記述されているが、ISO/IEC 7816-4 には SELECT FILE というカードコマンド名自体は規定されておらず SELECT だけとなっている。なお、FINEID の AID は「A0 00 00 00 63 50 4B 43 53 2D 31 35」が規定されているが、この AID は PKCS#15 を示したものとなっている。

両カードコマンドの違いとして、カードアプリケーションの選択に SELECT が用いられ、カード上のファイルの選択に SELECT FILE が用いられる。SELECT ではコマンドデータに AID を指定する。実際には、ISO/IEC 7816-4 でも AID を利用してカードアプリケーションを選択することが可能であり、その仕様は FINEID で指定しているカードコマンド内容と同じであることから、カードコマンドの APDU に関しては Open Platform の仕様を意識することなく、ISO/IEC 7816-4 だけを意識しておればよいと言える。

次に特徴的な点として、データの取得に関して READ BINARY と GET DATA を持つということが挙げられる。FINEID ドキュメント上の記述では、GET DATA に関し

ては RSA 公開鍵の取得時に使われるものとされている。後述するデータモデルでは、公開鍵は単独のファイルとしては存在しておらず、証明書ファイルに含まれているため、GET DATA で読み出すデータのファイルは証明書ファイルであることが考えられる。

MANAGE SECURITY ENVIRONMENT コマンドでは SET オペレーションと RESTORE オペレーションのみが規定されている。

PERFORM SECURITY OPERATION コマンドが持つ機能として、ISO/IEC 7816-8 では表 12 に示す 8 つのオペレーションを規定しているが、FINEID ではその内 HASH と COMPUTE DIGITAL SIGNATURE、DECIPHER の 3 つのみが規定されている。基本的にカード内に存在するプライベート鍵を利用したオペレーションのみが考慮されている。HASH コマンド自体はプライベート鍵を必要とするオペレーションではないが、COMPUTE DIGITAL SIGNATURE を行うときに入力値としてハッシュ化されていないデータも許容しているために、IC カード側でハッシュ値の計算を行うために用意されている。利用されるアルゴリズムは、HASH では SHA-1 のみが規定されており、COMPUTE DIGITAL SIGNATURE と DECIPHER では RSA 暗号のみが規定されている。RSA 暗号として、RSA プリミティブと PKCS#1v1.5 の暗号・電子署名に対応したオペレーションが可能である。

## データモデル

S4-1 では ISO/IEC 7816-15 に従い、FINEID のカード内情報の構成を示しているが、S4-1 の以前のバージョンは、ISO/IEC 7816-15 が策定される前であり、PKCS#15 に従った仕様となっている。その名残として、空き空間管理で利用される EF.UnusedSpace が使われており、これ自体は ISO/IEC 7816-15 では規定されていないデータモデルである。

プライベート鍵を 2 種類持つことは S4-1 で示されており、1 つが認証と暗号化用として、もう 1 つが否認防止の署名用として用いられる。双方とも RSA の 1024 ビット鍵である。2 つの RSA 鍵に従いカード保有者のエンドエンティティ証明書も 2 つ用意されている。証明書はその他に 2 つの証明書が入っており、これらはカード保有者の信頼点となる FINEID の国民登録局ルート CA と中間 CA の証明書である。またプライベート鍵にはそれぞれに対応した PIN が authId により結びついている。なお、証明書オブジェクトは X.509 証明書に公開鍵も含んでいる。FINEID における暗号情報オブジェクトの関連図を図 23 に、またファイルの構造図を図 24 に示す。

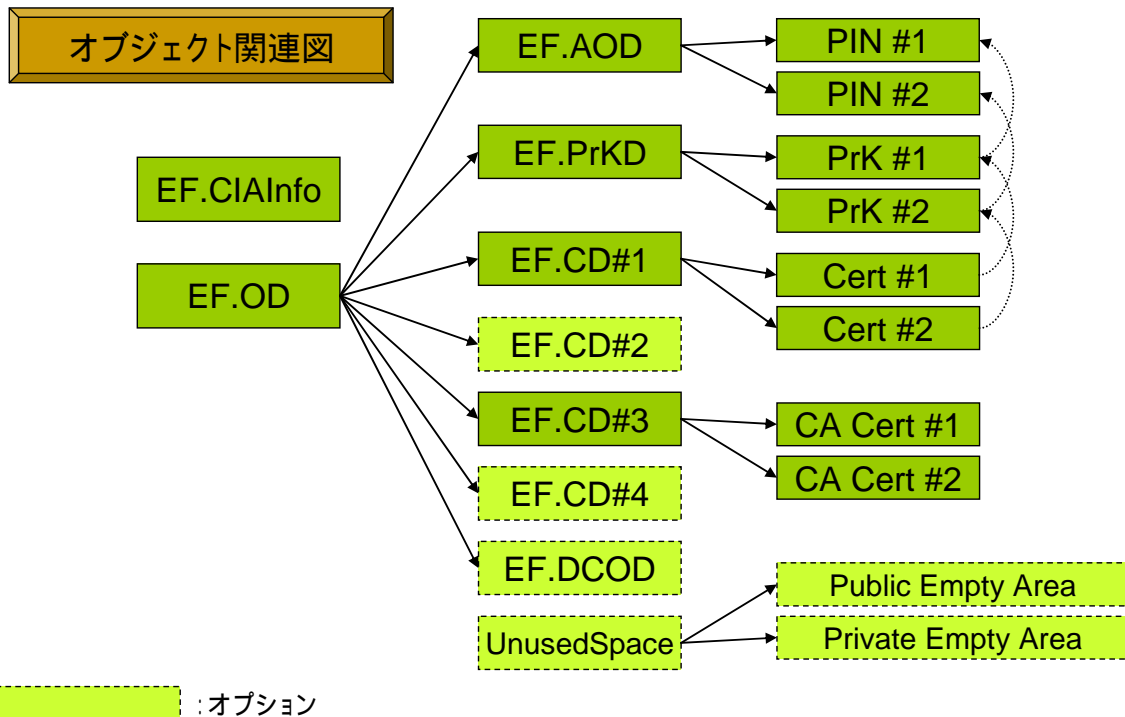


図 23 FINEID オブジェクト関連図

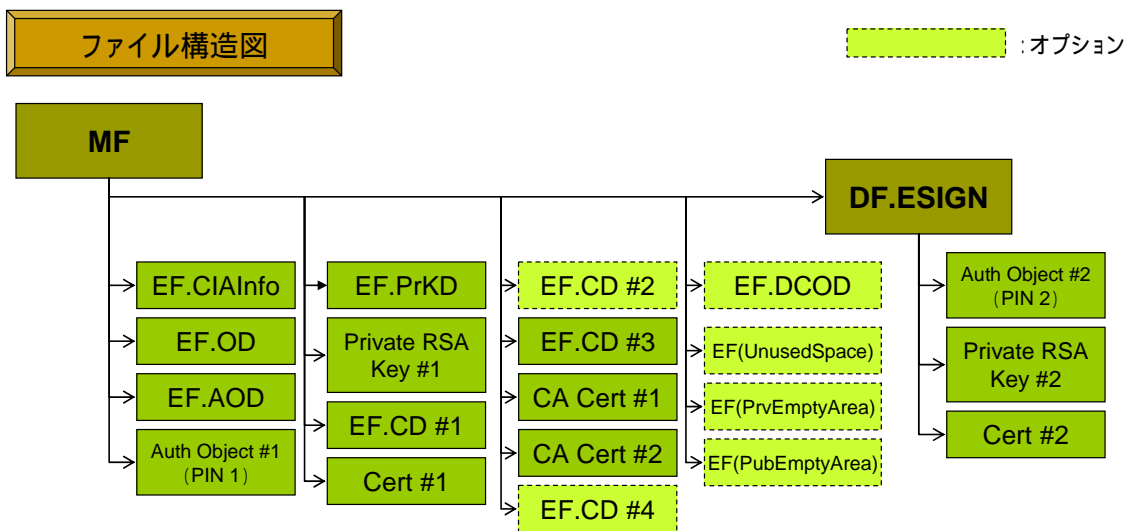


図 24 FINEID ファイル構造

データ構造の特徴的な点として、FINEID アプリケーションとは別に、DF.ESIGN と呼ばれるファイル、つまり別のカードアプリケーションが存在することである。このファイルの下には、電子署名用に用いられるプライベート鍵とその証明書、さらにプライベート鍵利用時に必要となる PIN が入っている。そしてこの DF には別途 AID として「A0 00 00 01 67 45 53 49 47 4E」が割り当てられている。この AID は欧州で別途進ん

でいる電子署名用の IC カードの仕様「安全な署名生成デバイスとしてのスマートカード用アプリケーションインターフェース (Application Interface for smart cards used as Secure Signature Creation Devices<sup>[37]</sup>)」(CWA 14890 シリーズ)におけるカードアプリケーションのものであり、FINEID カードを持つことでそれらにも対応可能なようになっている。この DF は FINEID のバージョンが 1 のころは規定されていなかったものであり、バージョンが 2 になってから規定されたものである。

EF.CD#3 は、国民登録局ルート CA の証明書と、その下位 CA であるカード保有者用の証明書発行を行った中間 CA の証明書の 2 つを入れるためのディレクトリである。2 つの証明書オブジェクトはそれぞれ authority の値が TRUE に設定されている。

EF.UnusedSpace は未使用領域を管理するためのディレクトリの役割を示すものであり、配下に EF.Public EmptyArea と EF.Private EmptyArea を持つ。なんらかのデータ追加や削除が起きた場合に、これら 2 つのファイルを利用して未使用領域を管理する。

なお、図 24 は、FINEID の仕様にある図から作成してのものであるが、仕様に載っている ISO/IEC 7816-15 で必須である DF.CIA が記述されていない。

## アクセス管理

S4-1 では PIN#2 の記述において、userConsent 要素が利用されるべき、としている。PIN#2 は否認防止の署名用のプライベート鍵を操作するときに必要な PIN である。S4-1 では userConsent に設定すべき値を明示していないものの、S1 での userConsent の説明においてその回数に触れている。ここでは、毎回の PIN 入力はセキュリティを高めるものの、利便性を損なうものとし、そのトレードオフに言及しており、その上で否認防止の電子署名に利用されるものであろう、としている。このことから、PIN#2 にあたっては userConsent が 1 に設定されていることを期待していることが考えられる。

また、セキュリティ環境に関して S1 では初期のセキュリティ環境の状態には触れていない。署名操作と復号操作に関する記述で、最初に SE を空 (Empty) にし、その後 MANAGE SECURITY ENVIRONMENT を SET、そして署名/復号操作を行うことが書かれている。このことから、利用されるカードコマンドに仕様において MANAGE SECURITY ENVIRONMENT コマンドでは SET と RESTORE オペレーションのみが規定されている。

### 3.3. BELPIC

#### 3.3.1. プロジェクト概要

BELPIC(Belgian Personal Identity Card)はベルギーの電子政府プロジェクトの一環として始まった国民 ID カードのプロジェクトであり、Belgium eID とも呼ばれる。プロジェクトは 2002 年末より開始されており、パイロットプロジェクトを経て、2005 年の 9 月からは新規に発行される ID カードがすべて BELPIC のカードとなっている。2009 年の末には 12 歳以上の国民への配布を終える見込みを立てており、2006 年 10 月の時点では 400 万枚を超えるカードが発行されている<sup>[22][39]</sup>。

PKI を利用したカードの用途は、基本的にはリモートからの電子認証と否認防止の署名の 2 つとなっている。しかし、いわゆる BELPIC カードは 12 歳以上の国民に配布されるものであり、12 歳未満の国民には kid@card というカードが配布されており、さらに年齢によりその用途も異なっている。表 22 に概要を示す。

表 22 ベルギーにおける IC・ID カード種類と格納証明書

年齢	カードの種類	証明書
0-6	kid@card	なし
6-12		認証用のみ
12-18	BELPIC カード	認証用と否認防止の署名用の 2 つ
18-		

ここで注目したいのは、18 歳以上の国民のみに否認防止の署名用の証明書が発行されていることである。否認防止の署名はすなわち、署名することに対する責任をあらわすものであり、ベルギー（あるいは BELPIC）においては 18 歳未満の国民はそういった責任能力をもたないということを示したのもであり、必要な発行対象のみに証明書を発行する好例と言えよう。

### 3.3.2. 技術概要

IC カードは Axalto 社の Cyberflex を利用している。Java VM の上に BELPIC の仕様を満足する Java アプレットが載っている。API として Java Card 2.1API と Global Platform 2.0.1 API が利用可能であり、PKCS#15 対応のデータが存在している。

また BELPIC ではカードを利用するためのミドルウェアも提供しているが、これは 4.1 節で解説する OpenSC を元にしたものである。

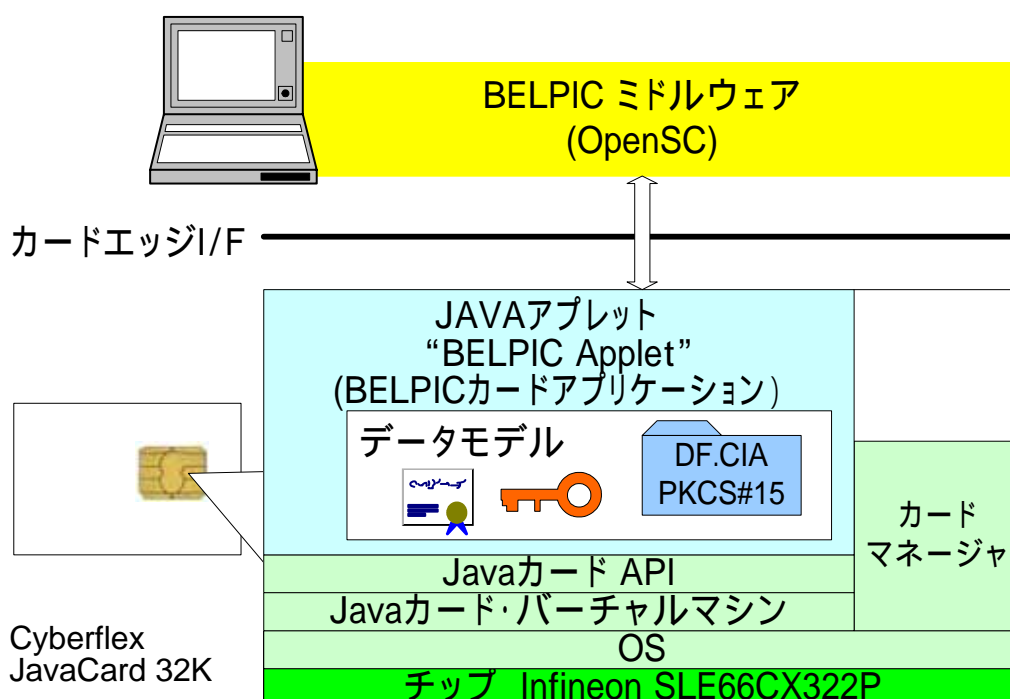


図 25 BELPIC カード概要

BELPIC は、FINEID と同様、カード保有者用に 2 つのプライベート鍵を用意している。それは認証用と否認防止の署名用の 2 つである。FINEID と違う点として、暗号用途の鍵を用意していないことがある。また、さらにいくつか別の公開鍵を持っている。鍵と証明書に関する詳細は、データモデルの部分で解説する。

BELPIC で利用される証明書のうち、署名用の証明書はクオリファイド証明書 (QC) である。CPS は公開されているが、CP は公開されていない。CPS には証明書プロファイルも含まれている。BELPIC で発行される証明書の信頼モデルは、BELPIC 単独のものではなく、ベルギーの電子政府全体における PKI の信頼モデルとなっている。図 26 にその信頼モデルを示す。

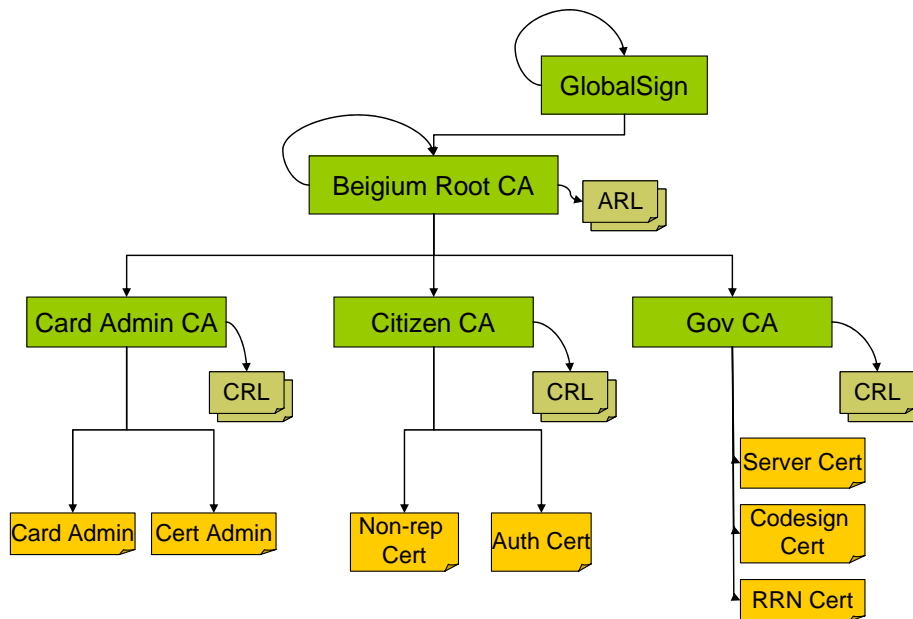


図 26 ベルギー政府 PKI の信頼モデル

カード内に格納されているものはベルギーのルート CA 自己署名証明書であり、IC・ID カード利用者の信頼の基点となっている。また、それとは別にルート CA は GlobalSign 社の CA より署名された CA 証明書の発行を受けている。GlobalSign の CA による署名を受けた証明書が存在する理由は、GlobalSign の CA が商用 CA でありすでに世の中の大部分の Web ブラウザに信頼点として登録されているためであり、そういった CA より証明書の発行をうけることで、下位 CA から発行したサーバ証明書などが利用可能となるためである。

他にカード内に保持している証明書が 2 つある。1 つは外部認証用に保持している CardAdmin CA の証明書である。もう 1 つとして、ベルギーの国民登録機関である RRN の証明書を持つ。RRN Cert のプライベート鍵により、カード発行時にカード内に含まれる個人の ID 情報や住所、顔写真データに署名が付されている。

### 3.3.3. カードコマンドとデータ構造、アクセス管理

#### カードコマンド

BELPIC で利用されるカードコマンドは基本的に ISO/IEC 7816-4 と 8 から構成され、2 つのコマンドが ISO/IEC 7816-9 より選択されている。表 23 に BELPIC で利用され

るコマンドの一覧を示す<sup>[40]</sup>。

表 23 BELPIC カードコマンド一覧

コマンド種別、コマンド名		参照標準
選択コマンド		
	SELECT FILE	ISO/IEC 7816-4
データ操作コマンド		
	READ BINARY	ISO/IEC 7816-4
	UPDATE BINARY	
	ERASE BINARY	
認証関連コマンド		
	VERIFY	
	CHANGE REFERENCE DATA	
	RESET RETRY COUNTER	
伝送処理関数		
	GET RESPONSE	ISO/IEC 7816-4
MANAGE SECURITY ENVIRONMENT		
	SET	
	RESTORE	
PERFORM SECURITY OPERATION		
	COMPUTE DIGITAL SIGNATURE	ISO/IEC 7816-8
	VERIFY DIGITAL SIGNATURE	
	VERIFY CERTIFICATE	
GENERATE PUBLIC KEY PAIR		
	ACTIVATE FILE	ISO/IEC
	DEACTIVATE FILE	7816-9

特徴的なのは ACTIVATE FILE、DEACTIVATE FILE の 2 つがあることであろう。

これら 2 つは ISO/IEC 7816-9 より選択されたカードコマンドである。これらの 2 つのカードコマンドにより、選択したファイルの状態を変化させる。

ファイルの選択用カードコマンドとして SELECT FILE、また公開鍵作成用のカードコマンドとして GENERATE PUBLIC KEY PAIR があるが、これらはいずれも BELPIC が参照している ISO/IEC 7816 シリーズが旧版のものであるためであり、実際は SELECT、GENERATE ASYMMETRIC KEY PAIR と変わりはない。

また VERIFY DIGITAL SIGNATURE コマンドが選択されていることも注目したい。IC カード自身の証明書を使って何らかのデータを検証するとき、IC カードで検証を行う必要はない。むしろ IC カードという限られたハードウェアリソースの中で検証する必要がないものは利用しないほうが良いといえる。しかしここで VERIFY DIGITAL SIGNATURE が選択されている理由は、BELPIC が外部認証を意識した仕様となっているからである。

また、FINEID と比較して、DECIPHER コマンドがないことにも注目したい。BELPIC では IC・ID カードの用途に暗号化を入れていないためである。

なお、BELPIC のカードアプリケーションを選択するとき用いられる AID は「A0 00 00 01 77 50 4B 43 53 2D 31 35」である。後半の 7 バイトが PKCS#15 と同じく ASCII コードで「PKCS-15」を示していることにも注目されたい。

## データモデル

BELPIC のデータモデルは、ISO/IEC 7816-15 ではなく PKCS#15v1.1 に準拠したものである。図 27 図 28 は BELPIC のデータモデルを示したものである。仕様上では EF.CIAInfo が EF (TokenInfo) と表現されているが、本報告書では ISO/IEC 7816-15 の表現に照らし合わせた表現に変更してあるために注意されたい。暗号情報オブジェクトの関連図を図 27 に示す。また、ファイル構造図を図 28 に示す。

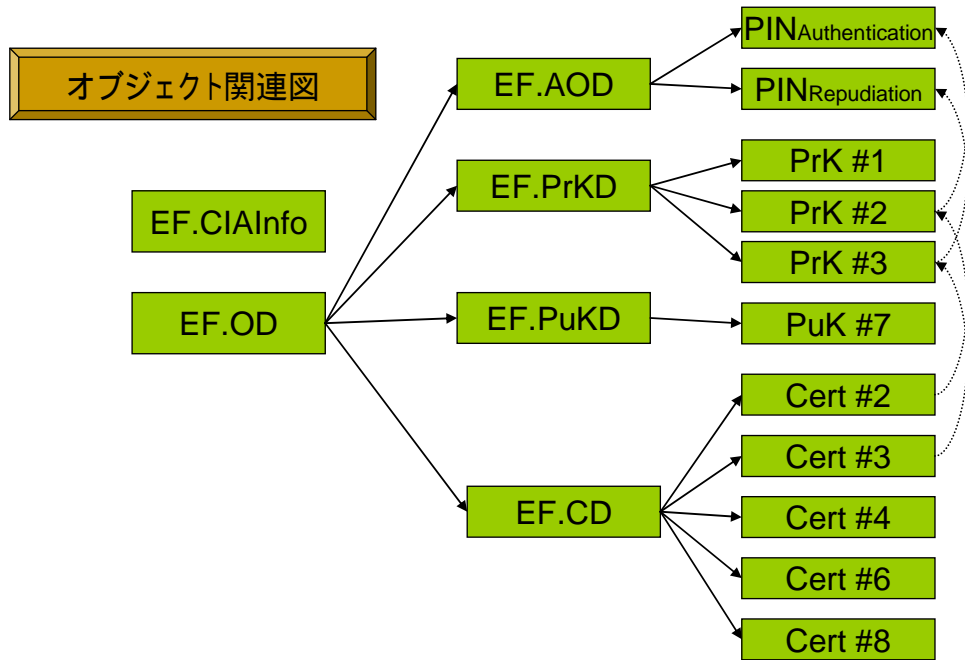


図 27 BELPIC オブジェクト関連図

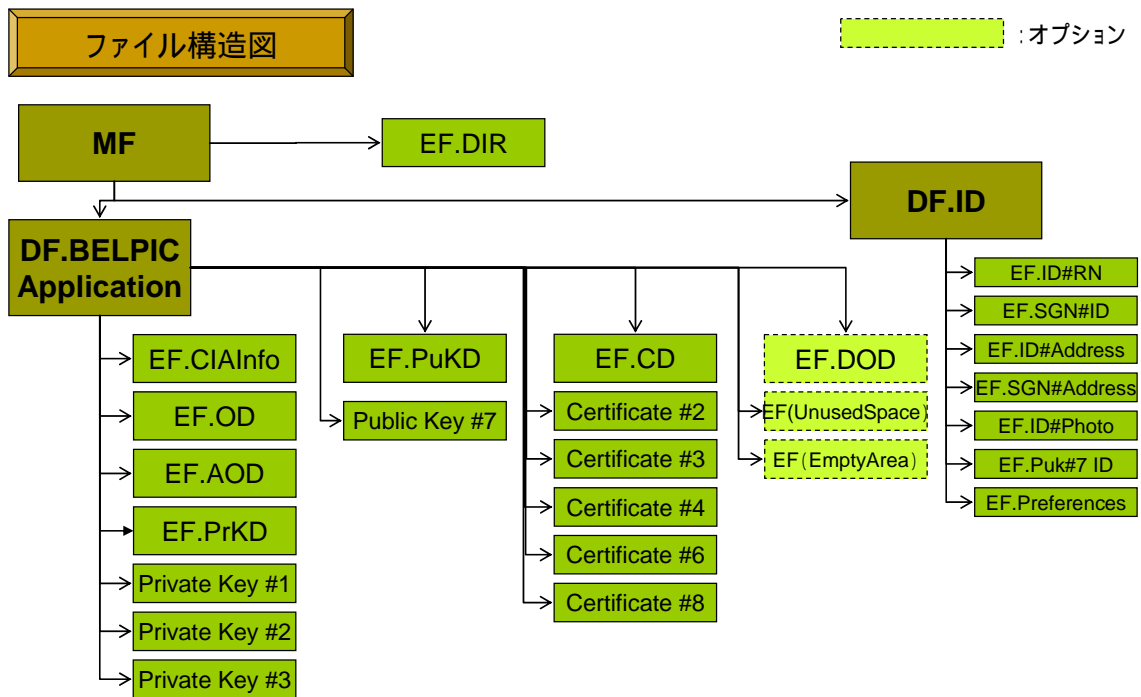


図 28 BELPIC ファイル構造図

BELPIC のファイル構造を見るとわかるように、BELPIC では 2 つのカードアプリケーションを搭載している。1 つが PKCS#15 のカードアプリケーションであり、もう 1 つはカード保有者である市民の情報が保管されているカードアプリケーションである。

図 28 のファイル構造で 2 つの DF によりそれらが分かれていることがわかる。

BELPIC は 2 つの鍵ペアと証明書をカード保有者である市民用として持っている他に、カードの認証用に 1 つのプライベート鍵を持っている。図 28 中の Private Key #1 がそれに当たる。認証用のプライベート鍵が Private Key #2、否認防止の署名用のプライベート鍵が Private Key #3 である。証明書の番号はプライベート鍵の番号に対応したものとなっている。Certificate #4 は、市民用の 2 つの証明書を発行した CA の証明書である。また Certificate #8 はベルギーの登録局 ( National Register : RRN ) の証明書である。DF.ID 下に保管されるいくつかの情報に登録局が電子署名をしており、その検証用のための証明書である。

一方、公開鍵ファイルとして Public Key #7 があるが、これは外部認証用の鍵である。この存在から、外部認証を用いてカード情報の更新などを行うことを考慮した仕様となっていることがわかる。

## アクセス管理

BELPIC では否認防止の署名用のプライベート鍵の操作について、userConsent の値を 1 に設定されていることが規定されている。

セキュリティ環境については、BELPIC は ISO/IEC 7816-4 と似たような記述をしているのみであり、その具体的な利用方法や初期の環境などについては触れていない。

## 3.4. PIV

### 3.4.1. プロジェクト概要

2004 年 8 月、米国では連邦政府施設への物理的・論理的アクセスのセキュリティ強化のため、身分証の標準を規定する大統領令 HSPD-12 が発令された。それに従い、米国立標準技術研究所 ( NIST ) が政府調達基準として FIPS ( Federal Information Processing Standard ) PUB 201 を発行し、身分証の標準を規定した。FIPS 201 のタイトルは「連邦従業員及び契約業者の個人識別情報の検証 ( Personal Identity Verification (PIV) of Federal Employees and Contractors<sup>[41]</sup> )」となっている。

PIV プロジェクトでは、身分証 IC カードの発行対象は連邦政府職員に加え、契約者 ( Contractor ) も含まれるために非常に多くの枚数が発行されることが見込まれており、

2009 年までに 2000 万枚が発行されると言われている<sup>1</sup>。

FIPS 201 は PIV のシステムとしての最小要件と詳細要件を定めているが、さらなる詳細な仕様に関しては幾つかの文書に分けて示している。また FIPS 201 は大きく 2 つのパートから構成されており、それぞれの PIV-I、PIV-II と呼んでいる。PIV-I は PIV を実現するにあたっての最小要件が示されている。PIV-II では、PIV-I で示された共通要件の目的に沿い、さらに連邦政府機関間での相互運用可能性を確保するための詳細な仕様を示されている。それら 2 つのパートの実現に関して、それぞれ時期を設定している。PIV-I に関しては 2005 年の 10 月 27 日までに全政府組織で達成されなければならないとし、2006 年の 10 月末には PIV-II が達成されなければならないとしている。各パートの詳細に関しては次節で説明する。

### 3.4.2. 技術概要

PIV カードは連邦政府の関係者が連邦政府機関への物理的なアクセス制御を行うことが主目的になっている。そのためカードの物理的なインターフェースとして接触・非接触の双方を考慮している。また、本章でこれまで解説してきた FINEID や BELPIC と異なっている点として否認防止の署名の利用がオプションになっていることがある。

さらに PIV で特徴的なのは、仕様に関わる文書の充実さであろう。基本となる標準として FIPS 201 があるが、仕様としてはそれだけでは十分ではなく、FIPS 201 に付随する形で複数の文書が仕様に関わる文書として規定されている。図 29 に PIV のカード・ミドルウェア・アプリケーションの関連図と、それらにかかわる仕様を示す<sup>[24][42]</sup>。

---

<sup>1</sup> この数字は NIST の Jim Dray 氏が 2006 年 7 月に NICSS の e-ID 講演会で講演した際に提示した数字であり、NIST の公式発表ではないことに注意されたい。

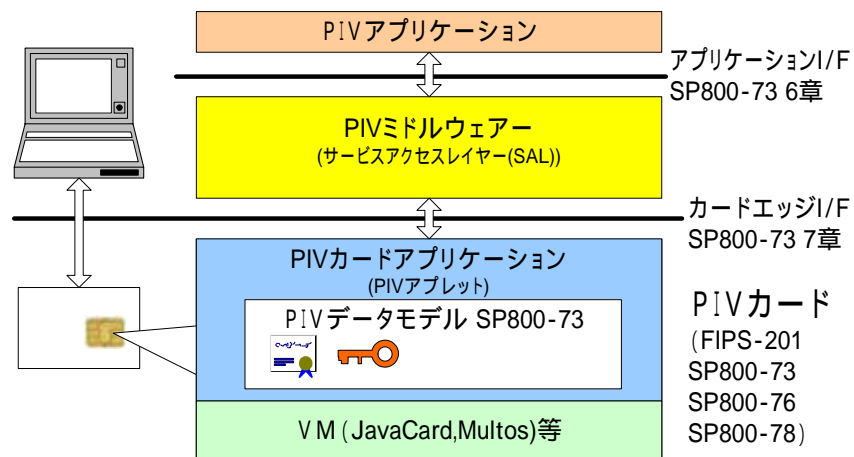


図 29 PIV カード概要と関連標準

本節で特に焦点を当てているカード内のデータモデルやカードコマンドインターフェース、セキュリティのアーキテクチャに関しては、NIST SP 800-73-1<sup>[21]</sup>に記述してある。次節では NIST SP 800-73-1 を中心に解説を進める。

### 3.4.3. カードコマンドとデータ構造、アクセス管理

SP 800-73-1 には、2 種類のインターフェースについて述べられている。1 つが移行期 (Transitional) インターフェースであり、もう 1 つがエンドポイントインターフェースである。これら 2 つのインターフェースは、これまでに米国連邦政府機関で利用されていた IC カードからの移行を考慮してのものである。たとえば米国の国防総省 (DoD) で利用されている IC カード CAC (Common Access Card) は既に 450 万枚もの発行がされており、こういった大量に発行された IC カードからの移行を考慮した仕様であることが PIV の大きな特徴とも言えよう。

移行期 (Transitional) のインターフェースに関しては、以前 NIST が公開した「連邦政府スマートカード相互運用性仕様 (Government Smart Card Interoperability Specification : GSC-IS<sup>[11]</sup>)」のサブセットとなっている。

なお、PIV カードアプリケーションを選択するときに用いられる AID は「A0 00 00 03 08 00 00 10 00 01 00」となっている。これはアプリケーション提供者識別子 (RID) として NIST の「A0 00 00 03 08」に続き、PIV カードアプリケーションを示すアプリケーション識別子 PIX「00 00 10 00」と、バージョンを示す「01 00」が示されている。

## カードコマンド

PIV で利用されるカードコマンドインターフェースは移行期 (Transitional) のものとエンドポイントのものに分かれる。ここではエンドポイントのカードコマンドのみ解説を行う。表 24 はエンドポイントインターフェースのカードコマンドを示す。

**表 24 PIV エンドポイントインターフェースのカードコマンド一覧**

コマンド種別、コマンド名		参照標準	接触	非接触
選択コマンド				
	SELECT	ISO/IEC 7816-4		
データ操作コマンド				
	GET DATA	ISO/IEC		
	PUT DATA	7816-4		×
認証関連コマンド				
	GENERAL AUTHENTICATE	ISO/IEC		
	VERIFY	7816-4		×
	CHANGE REFERENCE DATA			×
	RESET RETRY COUNTER			×
	GENERATE ASYMMETRIC KEY PAIR	ISO/IEC 7816 - 8		×

エンドポイントインターフェースのカードコマンドはすべて ISO/IEC 7816-4、あるいは 8 に規定されたそれぞれのカードコマンドの SLA と INS に従っている。しかし、データ操作コマンドである GET DATA と PUT DATA の 2 つに関しては、カードコマンドで入力するパラメータは ISO/IEC 7816-4 にないものであり、標準からは外れている。

これまでに解説した FINEID と BELPIC とのもっとも大きな違いとして、MANAGE SECURITY ENVIRONMENT コマンドと PERFORM SECURITY OPERATION コマンドを持たないことがある。これらのコマンドの代わりに、PIV では署名や暗号化のオペレーションを GENERAL AUTHENTICATE コマンドを使って実現している。

また PIV では接触・非接触の用途に応じて必要なコマンドを分けていることも注目したい。

## データモデル

カードコマンドインターフェースに関しては、PIV は ISO/IEC 7816-4 から外れた仕様ではあるものの基本的には標準仕様にのっとったものであった。一方で、データモデルに関しては ISO/IEC 7816-15 や PKCS#15 とは異なっており、そのまま使っているわけではないが、それらとの相互運用可能性を意識した部分もある。NIST が公開している報告書「PIV カード管理レポート (Personal Identity Verification Card Management Report<sup>[43]</sup>)」では ISO/IEC 7816-15 への対応についての記述があり、将来的な対応を視野にいれていることがわかる。

PIV のデータモデルは GSC-IS のデータモデルに従ったものとなっており、様々なデータを収めているものがファイルではなく、コンテナまたはオブジェクトと呼ばれるデータになっている。これらのコンテナにはそれぞれ独自のコンテナ ID が振られている。

表 25 PIV コンテナ ID 一覧

コンテナ名	コンテナ ID	必須/オプション
カード機能コンテナ	0xDB00	必須
CHUID バッファ	0x3000	必須
PIV 認証の証明書バッファ	0x0101	必須
指紋バッファ	0x6010	必須
印刷情報バッファ	0x3001	オプション
顔画像バッファ	0x6030	オプション
デジタル署名の証明書バッファ	0x0100	オプション
鍵管理の証明書バッファ	0x0102	オプション
カード認証の証明書バッファ	0x0500	オプション
セキュリティオブジェクトバッファ	0x9000	必須

コンテナの内部は各種データが TLV 形式で格納されている。たとえばカード機能コンテナの場合は以下の情報が格納される。

- カード識別子
- 機能コンテナバージョン番号
- 機能グラマーバージョン番号

- アプリケーション CardURL
- PKCS#15 対応
- 登録済みデータモデル番号
- アクセス制御ルールテーブル
- CARD APDU
- リダイレクトタグ
- 機能タブ (CT)
- ステータスタブ (ST)
- 次のカード機能コンテナ
- 拡張アプリケーション CardURL
- セキュリティオブジェクトバッファ
- エラー検出コード

実際にクライアントアプリケーション側から各データを呼び出す際には、PIV はオブジェクト識別子 (OID) を利用して選択する。表 26 に OID を示す。

表 26 PIV データオブジェクトと OID

データオブジェクト名	OID	BER-TLV タグ
カード機能コンテナ	2.16.840.1.101.3.7.1.219.0	0x5FC107
CHUID	2.16.840.1.101.3.7.2.48.0	0x5FC102
PIV 認証の証明書	2.16.840.1.101.3.7.2.1.1	0x5FC105
指紋情報	2.16.840.1.101.3.7.2.96.16	0x5FC103
印刷情報	2.16.840.1.101.3.7.2.48.1	0x5FC109
顔画像	2.16.840.1.101.3.7.2.96.48	0x5FC108
デジタル署名の証明書	2.16.840.1.101.3.7.2.1.0	0x5FC10A
鍵管理の証明書	2.16.840.1.101.3.7.2.1.2	0x5FC10B
カード認証の証明書	2.16.840.1.101.3.7.2.5.0	0x5FC101
セキュリティオブジェクト	2.16.840.1.101.3.7.2.144.0	0x5FC106

## アクセス管理

カード内情報へのアクセス権限に関して、PIV ではアクセス制御ルールやセキュリティステータスという情報を持つ。たとえば鍵へのアクセスに関して、PIN による認証が

されると鍵へのアクセスが可能になるようにセキュリティステータスが変更される、などがある。

否認防止用の署名時での PIN 入力を制御する userConsent に対応するものは PIV には詳細には定義されていないが、否認防止用の署名鍵に関する記述に、電子署名を行うたびにカード保有者の確認が必要である旨が書かれている。

また外部認証に関しては、SP 800-73-1 に特筆されていないものの、エンドポイントインターフェースのカードコマンドでの GENERAL AUTHENTICATION で外部認証が可能である。しかし、それ以外の移行期 (Transition) インターフェースでは利用不可能となっており、現在のバージョンでは外部認証を考慮していないことがうかがえる。

#### 3.4.4. PIV カードアプレットの参考実装

Java アプレットとは、JavaVM を搭載した Java カード上で稼動する小さなプログラムのことで、カードエッジ・インターフェースを実現する。PIV カードエッジ・インターフェースを実現するための Java アプレットの実装としては、さまざまなものが考えられるが、ここではミシガン州立大学のスマートカード・チームが作成した、PIV のカードエッジ・インターフェース仕様を実装した Java カードアプレットプログラムの実装について解説する。なおこのアプレットプログラムは BSD ライセンスの元でオープンソースソフトウェアとして公開されている。  
(<http://www.identityalliance.com/downloads/PIV-II-MSU.zip>,  
<http://lists.musclecard.com/pipermail/muscle/2005-September/004387.html>)

このプログラム実装はコンパクトで、javacard.framework.Applet である CardEdge クラス(1010 ステップ)と、データとそのアクセス制御リストを管理するための DEMS (Data Element Management System:492 ステップ)クラスの2つから構成されている。PIV カードコマンドをハンドルするのは CardEdge クラスであり、DEMS クラスはデータの内部表現であるので、ここでは主に CardEdge クラスについて解説する。付属ドキュメントによると、アプレット作成作業が PIV の仕様策定と同時期に行われたため、コマンドチェーンの実装、GET DATA カードコマンドが仕様と異なり、GENERAL AUTHENTICATE による MUTUAL AUTHENTICATE が完全には実装されていないという注意書きがある。

一般に Java カードアプレットは install,select,process の3つのメソッドを持ち、このメソッドがフレームワークからコールされることで、動作を行う。

表 27 Java カードアプレットのメソッド

メソッド	説明
public static void install(byte[] bArray, short bOffset, byte bLength) throws ISOException	アプレットインスタンスの生成、メモリ割り当ての取得などの初期化作業を行う。
public boolean select()	IC カード内で、このアプレットが選択された（SELECT カードコマンドでこのアプレットの AID が指定された）とき、呼ばれる。
public void process(APDU apdu) throws ISOException	外部から APDU の送信があったときにこのメソッドが呼ばれる。引数の APDU をみて、自身の動作を決定し、APDU にデータをセットすることで、応答する。エラーを発生させるときは ISOException をスローする。

### install

参考実装の install メソッドでは、CardEdge インスタンスを生成しているのみで、特にメモリ領域の確保などは行わない。CardEdge クラスが DEMS インスタンスを保持していて、DEMS クラス内部でメモリを確保している。

### select

参考実装の select メソッドは何もしない。

### process

このメソッドが、重要な位置を占めている。受信した APDU からインストラクションバイト（INS）を取得し、その値を元に switch-case で動作を決定している。その中で PIV 仕様にはない GetAdminKey、AddPin という動作を実装しているが、これらについては特に解説しない。

表 28 カードエッジコマンドと開発量

コマンド種別、コマンド名	参照標準	開発量	メモ
選択コマンド			
SELECT	ISO/IEC 7816-4	19 行	送られてきた APDU を確認するのみで、特に処理は行っていない。

データ操作コマンド				
	GET DATA	ISO/IEC 7816-4	52 行	送られてきた APDU からデータの種別を判別し、送り返している。おもに応答データが 255 バイトを超えた場合の処理にコード量をとられている。
	PUT DATA		43 行	データの書き換え、または新規追加を行う。書き換え対象データへのアクセス権限があるかチェックしている。
認証関連コマンド				
	GENERAL AUTHENTICATE	ISO/IEC 7816-4	303 行	もっとも大きなメソッド。しかし、付属ドキュメントにあるように一部正しい実装がなされていない。メソッドは内部でさらに処理が分かれている。詳細は後述する。
	VERIFY		36 行	渡された鍵参照が 0x00 (カード保有者グローバル PIN) 以外の場合エラーとしている。
	CHANGE REFERENCE DATA			未実装。このカードコマンドは PIN の変更を行う際に使用するものだが、本実装では、一度設定された PIN は変更できない簡易な実装となっている。
	RESET    RETRY COUNTER		38 行	鍵のブロック解除を行う際、カード保有者の PIN による認証を行っている。PIV 仕様によるとこのコマンドを実行するにはアプリケーション管理者と、カード所有者のバイOMETリックによる認証が必要とされている。このアプレットでは簡易な実装になっている。

GENERATE ASYMMETRIC KEY PAIR		75 行	鍵の新規生成のみサポートしており、すでにある鍵参照の置き換えはできない簡易な実装となっている。認証の機構はない。
---------------------------------	--	------	--

## GENERAL AUTHENTICATE

このカードコマンドの実装部分は CHALLENGE を送信する部分 ( GET CHALLENGE に相当 ) クライアントから取得した暗号データを複合し、先に送信した CHALLENGE と比較する部分 ( EXTERNAL AUTHENTICATE に相当 ) クライアントから送られてきたデータを暗号化して送り返す部分 ( INTERNAL AUTHENTICATE に相当 ) と、テストされていない MUTUAL AUTHENTICATE 部分から構成されている。

アプレットプログラムは GENERAL AUTHENTICATE カードコマンドを受信した際、カードコマンドの引数の内容と、現在のカードの状態がチャレンジを送信した後かどうかによってどのような処理を行うか判定している。

### 3.4.5. PIV の今後の拡張

現在の PIV の仕様には暗号情報を探し出す仕組み ( Cryptographic Object Discovery ) が含まれていない。従って、ミドルウェアは独自の方法でオブジェクトを探し出す仕組みを実装しなければならない。必須でない暗号オブジェクト、例えば管理者証明書がカードに入っているか どうかを調べるには、ミドルウェアは GET DATA を実行してデータの取得を試みるしか方法がない。

NISTIR 7284 "Personal Identity Verification Card Management Report" では ISO7816-15 の概要と、例として FINEID や BELPIC などをあげ、PIV が将来拡張されてゆく段階では、カード上に暗号オブジェクトに関する情報を保持し、効率的に暗号情報オブジェクトを見つけられるようにする機構が不可欠であると述べている。また、SP800-73 が規定するクライアント API にはカードの発行、パーソナライズなどを行う際に必要となるカード管理に 関する仕様が含まれていないことについても言及しており、それらの機能を持つ、高度な API を整備することによってカードの発行、管理についても認定プロセスにのせることができるようになり、より高い 一貫性をもてると述べている。

### 3.5. カードコマンド、データモデルの比較

本節では、これまでに解説してきた標準や事例におけるカードコマンドとデータモデルの比較を行う。

#### 3.5.1. カードコマンドの比較

ここでは表 8 にもとづいて、各標準や事例の仕様がどういったカードコマンドをサポートしているかを示す。表 29 にそれらのサポート状況を示す。

表 29 各標準・仕様で利用されているカードコマンドの比較

コマンド種別、コマンド名	ISO/IEC 24727-2	FINEID	BELPIC	PIV
選択コマンド				
SELECT				
MANAGE CHANNEL				
データ操作コマンド				
READ BINARY				
WRITE BINARY				
UPDATE BINARY				
SEARCH BINARY				
ERASE BINARY				
READ RECORD				
WRITE RECORD				
UPDATE RECORD				
APPEND RECORD				
SEARCH RECORD				
ERASE RECORD				
GET DATA				
PUT DATA				
認証関連コマンド				
INTERNAL AUTHENTICATE				

	GET CHALLENGE				
	EXTERNAL AUTHENTICATE				
	GENERAL AUTHENTICATE				
	VERIFY				
	CHANGE REFERENCE DATA				
	ENABLE VERIFICATION REQUIREMENT				
	DISABLE VERIFICATION REQUIREMENT				
	RESET RETRY COUNTER				
伝送処理関数					
	GET RESPONSE				
	ENVELOPE				
MANAGE SECURITY ENVIRONMENT					
	SET				
	STORE				
	RESTORE				
	ERASE				
PERFORM SECURITY OPERATION					
	COMPUTE CRYPTOGRAPHIC CHECKSUM				
	COMPUTE DIGITAL SIGNATURE				
	HASH				
	VERIFY CRYPTOGRAPHIC CHECKSUM				
	VERIFY DIGITAL SIGNATURE				
	VERIFY CERTIFICATE				
	ENCIPHER				

	DECIPHER				
GENERATE ASYMMETRIC KEY PAIR					

これらから、各仕様の特徴や共通点が見えてくる。まず PIV が MANAGE SECURITY ENVIRONMENT や PERFORM SECURITY OPERATION のコマンドを持たないことが他の 3 つと異なる大きな特徴となっている。MANAGE SECURITY ENVIRONMENT コマンドに関しては、セキュリティ環境と同等のものを PIV では GENERAL AUTHENTICATE コマンドの引数として直接データを指定しやりとりしているために、コマンドとして必要としていないことが考えられる。PERFORM SECURITY ENVIRONMENT コマンドに関しては、暗号・署名といった操作をすべて GENERAL AUTHENTICATE コマンドで行っている。

その他 3 つの 24727-2、FINEID、BELPIC に関する特徴点として注目したいのは、MANAGE SECURITY ENVIRONMENT コマンドである。これら 3 つはすべてが SET オペレーションと RESTORE オペレーションのみの規定となっている。FINEID では RESTORE オペレーション時に空 (Empty) にして、各コンポーネントを SET オペレーションで再度設定するような運用を考慮しているが、他の 2 つでも同様の運用が考慮されていることが十分に考えられる。

また PERFORM SECURITY OPERATION に関しては、FINEID と BELPIC の差異に関しては、それぞれの ID カードの用途が異なることがオペレーションの差になっている。FINEID と比較して、BELPIC では暗号化を用途に入れていない。そのため DECIPHER オペレーションは規定されていない。一方で、FINEID は BELPIC と比較して、外部認証を考慮に入れていない。そのために FINEID では VERIFY DIGITAL SIGNATURE と VERIFY CERTIFICATE が規定されていない。そして、24727-2 ではこれら 2 つの仕様を包含するようにコマンドが規定されている。

### 3.5.2. データモデルの比較

本節では ISO/IEC 7816-15 あるいは PKCS#15 に対応した FINEID と BELPIC の 2 つにおけるデータオブジェクトの比較を行う。PIV に関しては PKCS#15 に準拠していないために本節では省略した。

表 30 FINEID と BELPIC のデータモデル比較

オブジェクト	FINEID		BELPIC
DF.CIA			< 規定なし >
EF.CIAInfo			
EF.OD	< 必須オブジェクト >		< 必須オブジェクト >
EFAOD	2 つの PIN への参照を持つ		2 つの PIN への参照を持つ。
EF.PrKD	2 つのプライベート鍵への参照を持つ		3 つのプライベート鍵への参照を持つ。
EF.PuKD	< 規定なし >		1 つの公開鍵への参照を持つ。
EF.CD	4 つの証明書オブジェクトを持つ(下記参照)		証明書オブジェクトは 1 つ。5 つの証明書への参照を持つ。
	EF.CD#1	カード保有者の証明書への参照を持つ	
	EF.CD#2	オプションで利用可能	
	EF.CD#3	カード保有者証明書の発行に関連する CA 証明書への参照を持つ	
	EF.CD#4	オプションで利用可能	
EF.UnusedSpace	オプションで利用可能		

もっとも大きな違いは EF.CD の構造であろう。FINEID では証明書の区分によりディレクトリとなる EF.CD を複数用意しているのに比べて、BELPIC では 1 つの EF.CD でさまざまな証明書を含んでいる。

また、BELPIC は FINEID にはない EF.PuKD を持つが、これは BELPIC が外部認証を意識した仕様であり、外部認証のための公開鍵を持つことの差から来るものである。

## 4. ミドルウェアの実装例

本章では IC・ID カードサービスを構成する一部分である、ミドルウェアの実装例として各種 IC・ID カードを扱うためのライブラリをオープンソースとして提供する OpenSC<sup>[44]</sup>と、3.4 節で取り上げた PIV について解説する。

OpenSC は本報告書作成時に最新であるバージョン 0.11.1 について解説する。

### 4.1. ミドルウェアの実装(OpenSC)

#### 4.1.1. OpenSC プロジェクトの概要

OpenSC プロジェクトは IC・ID カード等を扱うためのツール、ライブラリを開発しているオープンソースプロジェクトで、多くのサブプロジェクトにより構成される。

表 31 OpenSC のサブプロジェクト

サブプロジェクト名	概要	開発規模
OpenSC	メインプロジェクト。Linux,Mac OS, X-Windows 上で稼動する、汎用的なカード用ライブラリ、PKCS#11 ライブラリ、ツールなど	9 万ステップ
Windows Installer SCB	Windows 用バイナリパッケージと、Windows 独自のツールをサポートする	各種ツールをコンパイルし、まとめたもの。
Apple Mac OS X Installer SCA	Mac OS X 用バイナリパッケージと、Mac ネイティブアプリケーションサポート用プログラム	各種ツールをコンパイルし、まとめたもの。
OpenSSL PKCS#11 Engine	OpenSSL 用 PKCS#11 プログラム。OpenSSL で IC・ID カードを利用できる	1 千ステップ
GTK Card	IC・ID カード操作の GUI アプリケーション	8 千ステップ
OpenCT	カードリーダー、USB トークン用のドライバ	3 万ステップ

Pam PKCS#11	高機能 PAM 認証モジュール	1 万 3 千ステップ
Pam P11	単機能 PAM 認証モジュール	1 千ステップ
Libp11	PKCS#11 ライブラリ	6 千ステップ
OpenSC-Java	Java1.5 用の IC・ID カードサポート	1 万 1 千ステップ

本報告書では汎用的な IC・ID カードサポートライブラリ、ツールを開発する OpenSC について報告する。

OpenSC の中心は、IC・ID カードにアクセスするための libopensc ライブラリである。libopensc は、IC・ID カードだけでなく USB トークンもサポートする。基本プラットフォームは、Linux と Mac OS X、Windows である。OpenSC でサポートするカード、USB トークンを以下に挙げる。

表 32 OpenSC がサポートする IC カード等

カード名	説明
国等が発行している ID カード	
フィンランド ID Card FINEID	PKCS#15 準拠カード カード OS は、Setec Setcos
スウェーデン Posten eID	PKCS#15 準拠カード スウェーデンの郵便局の eID
エストニア ID Card EstEID	pkcs15-esteid.c(データモデル) 100 万枚以上の ID カードを発行
イタリア Infocamere	pkcs15-infocamere.c(データモデル) イタリアの民間認証局のカード
イタリア Postecert	pkcs15-postecert.c(データモデル) イタリア郵政局の事業会社である Postecom 社が発行
ベルギー-BELPIC	PKCS#15 準拠カード 2006 年 10 月現在 400 万枚以上のカードを発行。ミドルウェアに OpenSC を全面的に採用
スペイン Ceres	PKCS#15 準拠カード発行 スペインの電子 ID カード
ドイツ ID Cards, eHBA,	pkcs15-tcos.c(データモデル)

eGK	ドイツポストの認証局(SignTrust)が発行するカード。ドイツテレコム(Telesec)が発行するカード
台湾	PKCS#15 準拠カード 台湾市民 IC カード
オーストリア Bürgerkarte, e-card	pkcs15-atrust-acos.c(データモデル) オーストリアの市民カード
米国 PIV card applet	pkcs15-postecert.c(データモデル) card-piv.c(カードエッジ I/F)
カード OS(カードエッジ I/F)	
Schlumberger/Axalto Cryptoflex	card-flex.c(カードエッジ I/F)
Gemplus GPK	card-gpk.c(カードエッジ I/F)
EMV	card-emv.c(カードエッジ I/F)
Siemens CardOS M4	card-cardos.c(カードエッジ I/F)
IBM JCOP	card-jcop.c(カードエッジ I/F)
Micardo	card-mcrd.c(カードエッジ I/F)
Oberthur	card-oberthur.c(カードエッジ I/F)
OpenPGP	card-openpgp.c(カードエッジ I/F)
Setec Setcos	card-setcos.c(カードエッジ I/F)
Giesecke & Devrient Starcos	card-starcos.c(カードエッジ I/F)
Giesecke & Devrient Seccos	
TCOS based cards (NetKey E4, SignTrust, Smartkey)	card-tcos.c(カードエッジ I/F)
USB トークン	
Aladdin eToken Pro	card-cardos.c Siemens CardOS M4
Eutron CryptoIdentity ITSEC	card-starcos.c G&D 社の Starcos
Schlumberger/Axalto e-gate	card-flex.c Schlumberger/Axalto Cyberflex
Rainbow iKey 3000	card-starcos.c G&D 社の Starcos

#### 4.1.2. 全体のアーキテクチャ

OpenSC プロジェクト内の OpenSC サブプロジェクトは、プロジェクト全体の中で中心的な役割を担うライブラリである Smart Card Library (libopenc) と、それを利用したいいくつかのツール、その他のライブラリを提供する。

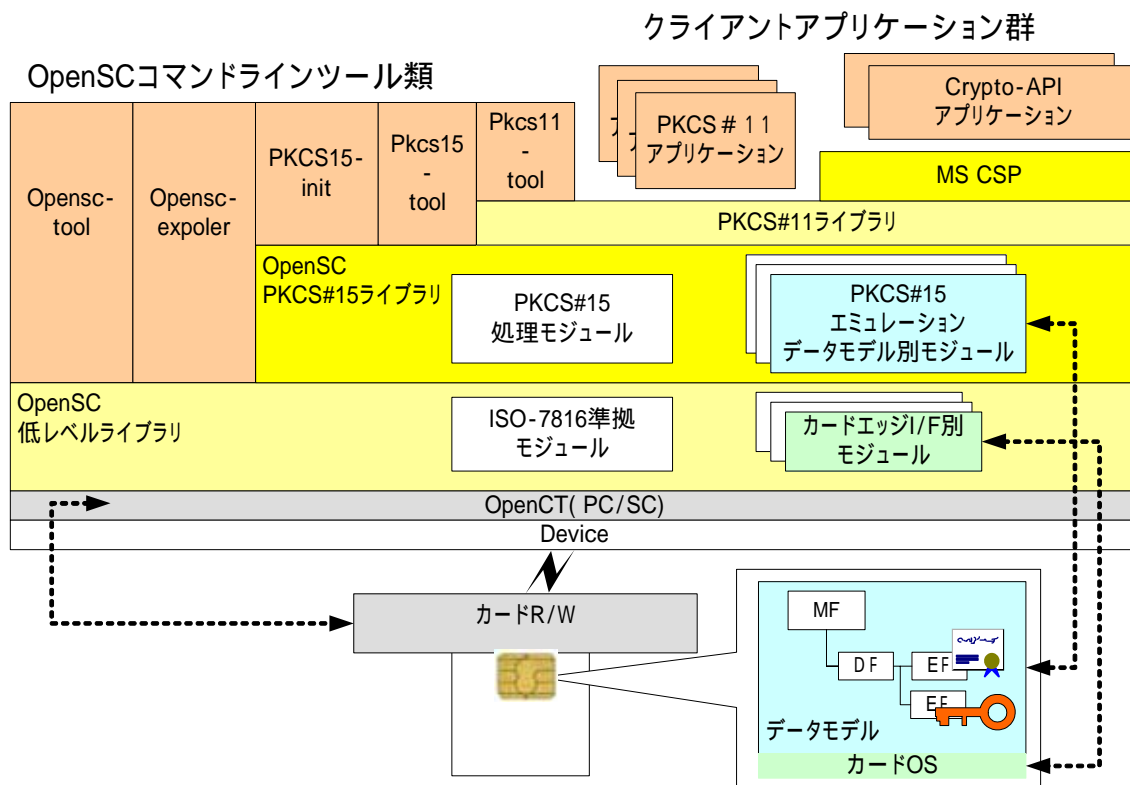


図 30 OpenSC のアーキテクチャ

実際にカードと通信をするカードリーダー/ライタを制御するレイヤーには、広く利用されている PC/SC や、OpenSC プロジェクトの一部である OpenCT、比較的古いハードウェアをサポートするための CT-API が利用できる。

その上に OpenSC 低レベルライブラリが位置し、カードコマンドレベルの操作を行う。OpenSC 低レベルライブラリには、カードエッジインターフェース毎のコマンド実装の違いを吸収するために、カードエッジインターフェース・別モジュールを動作させる機構を持つ。

OpenSC 低レベルライブラリを利用する形で OpenSC PKCS#15 ライブラリがあり、カードコマンドレベルではなく、もう少し高いレベルの暗号、署名といった操作を提供している。OpenSC PKCS#15 ライブラリは、カードが PKCS#15 準拠である場合、標

準実装の関数を使ってデータを読み込み、そうでない場合、PKCS#15 エミュレーション IC・ID カード別モジュールにハードコードされた情報（どのファイルに何が格納されているか、鍵と PIN の対応など）を使って、データを読み込む。

PKCS#11 ライブラリは、OpenSC PKCS#15 ライブラリの上位に実装され、OpenSC を利用するクライアントアプリケーションに対して、PKCS#11 インタフェースを提供する。

OpenSC は、提供するライブラリ機能を実行したり、カードを PKCS#15 フォーマットで初期化するためのツールなども提供する。

### コマンドラインツール類 / その他のライブラリ

OpenSC では、IC・ID カードに対して、基本的な操作を行うためのいくつかのコマンドラインツールを用意している。

表 33 OpenSC のコマンドラインツールなど

コマンドラインツール	説明
opensc-tool	カードの個別機能に依存しない機能を利用するためのツール。APDU の送信や、ファイルの一覧表示などを行うことができる
opensc-explorer	カード内のデータ（主にファイル）にアクセスするためのインタラクティブなツール
pkcs15-init コマンド	カードを PKCS#15 準拠フォーマットで初期化するためのツール
pkcs15-tool	PKCS#15 データオブジェクトを操作するためのツール
pkcs#11 ライブラリ	PKCS#11 のライブラリ。クライアントアプリケーションから利用する。内部で OpenSC PKCS#15 ライブラリを呼び出している
pkcs11-tool	PKCS#11 ライブラリを使用し、カードを操作するためのツール
CSP#11	Windows の CSP。これは OpenSC プロジェクトで作成されたものではないが、このツールを使用すると Windows の CryptoAPI から

OpenSC を使用できるようになる。

#### 4.1.3. カードエッジインタフェース

OpenSC 低レベルライブラリは、カードリーダー/ライタを制御するレイヤーの上に位置し、カードとの通信をカードコマンドレベルで行う。OpenSC 低レベルライブラリが提供する機能は、ISO/IEC-7816-4<sup>[8]</sup>,8<sup>[29]</sup>,9<sup>[30]</sup>で規定される機能である。

このライブラリはカードエッジインタフェースに依存しており、基本実装ではカードが ISO/IEC7816 準拠カードであり、SELECT,VERIFY,READ BINARY のほか、いくつかのカードコマンドが実装されていることを前提としている。カードが ISO/IEC7816 準拠ではない場合や、OpenSC 低レベルライブラリが必要とするカードコマンドを実装していない場合はカードごとにカードエッジインターフェース・モジュールを用意して、その違いを吸収している。

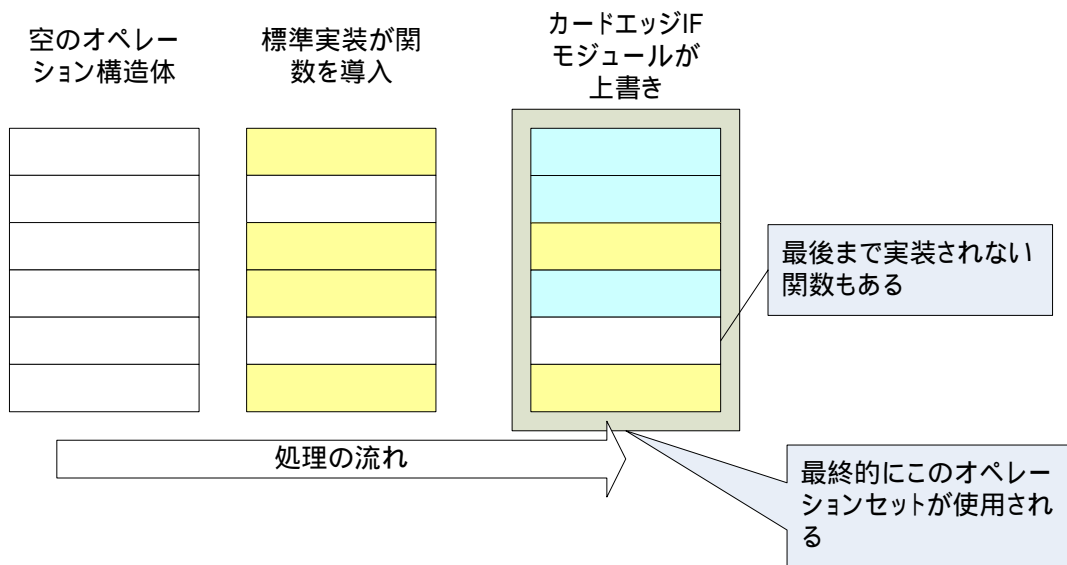


図 31 オペレーション構造体にオペレーションをセットする

OpenSC 低レベルライブラリが、カードエッジインタフェース部分、各種カード OS の ISO/IEC7816-4,8,9 レベルの違いを吸収している。ISO/IEC7816-4,8,9 で定められた多くのカードコマンドをすべてサポートしているわけではない。その中から OpenSC 低レベルライブラリが使用するコマンドを列挙して、それらコマンドに対応する関数をまとめた構造体 `sc_card_operations` を定義している[`opensc.h`]。その構造体には、初期値

として iso7816 標準実装関数がセットされる [iso7816.c]。各カードエッジインタフェースに対応した専用モジュールは、その値を上書きする。OpenSC 低レベルライブラリは、常に sc\_card\_operations に格納された関数をコールするため、専用の関数が呼び出されるようになる。

```
static struct sc_card_operations iso_ops = {
    no_match,
    NULL,          /* init */
    NULL,          /* finish */
    iso7816_read_binary,
    iso7816_write_binary,
    iso7816_update_binary,
    NULL,          /* erase_binary */
    iso7816_read_record,
    iso7816_write_record,
    iso7816_append_record,
    iso7816_update_record,
    iso7816_select_file,
    iso7816_get_response,
    iso7816_get_challenge,
    NULL,          /* verify */
    iso7816_logout,
    iso7816_restore_security_env,
    iso7816_set_security_env,
    iso7816_decipher,
    iso7816_compute_signature,
    NULL,          /* change_reference_data */
    NULL,          /* reset_retry_counter */
    iso7816_create_file,
    iso7816_delete_file,
    NULL,          /* list_files */
    iso7816_check_sw,
    NULL,          /* card_ctl */
    iso7816_process_fci,
    iso7816_construct_fci,
    iso7816_pin_cmd,
    NULL,          /* get_data */
    NULL,          /* put_data */
    NULL,          /* delete_record */
};
```

この構造体の一番初めの match\_card 関数( no\_match 関数がセットされている ) init 関数、finsh 関数は、ISO/IEC7816 のコマンドとは関係がなく、OpenSC が内部的に使用する関数である。

個別のカードエッジインタフェースを必要とする例として、Axalto 社製の IC カードである Cryptoflex カード ( Cryptoflex 8K,16K,32K v4,32K e-gate,Multiflex,Cyberflex など ) がある。これらをサポートするコード [card-flex.c] では、16 の関数が置き換えられており、これにより置き換わった関数については、iso7816.c に定義されているもの

ではなく、card-flex.c によって定義されたものが実行されるようになる。

```
struct sc_card_driver * sc_get_cyberflex_driver(void)
{
    if (iso_ops == NULL)
        iso_ops = sc_get_iso7816_driver()->ops;

    cyberflex_ops = *iso_ops;
    cyberflex_ops.match_card = cyberflex_match_card;
    cyberflex_ops.init = flex_init;
    cyberflex_ops.finish = flex_finish;
    cyberflex_ops.process_fci = cyberflex_process_file_attrs;
    cyberflex_ops.construct_fci = cyberflex_construct_file_attrs;
    cyberflex_ops.select_file = flex_select_file;
    cyberflex_ops.list_files = cyberflex_list_files;
    cyberflex_ops.delete_file = flex_delete_file;
    cyberflex_ops.create_file = flex_create_file;
    cyberflex_ops.card_ctl = flex_card_ctl;
    cyberflex_ops.set_security_env = flex_set_security_env;
    cyberflex_ops.restore_security_env = flex_restore_security_env;
    cyberflex_ops.compute_signature = cyberflex_compute_signature;
    cyberflex_ops.decipher = flex_decipher;
    cyberflex_ops.pin_cmd = flex_pin_cmd;
    cyberflex_ops.logout = flex_logout;
    return &cyberflex_drv;
}
```

#### 4.1.3.1. カードエッジインタフェース・モジュール切り替えの仕組み

設定によって強制的に指定された場合は、OpenSC 低レベルライブラリは指定されたカードエッジインタフェース・モジュールを使用するが、そうでない場合は、複数ある組み込み済みのカードエッジインタフェース・モジュールの match\_card 関数を順番に呼び出し、true を返したカードエッジインタフェース・モジュールを使用する実装になっている。どのカードエッジインタフェース・モジュールを使用するかは、カードエッジインタフェース・モジュールが match\_card 関数をどのように実装しているのかに依存するが、ほとんどのカードエッジインタフェース・モジュールは、カードが返した ATR (answer to reset) をみて、自分が処理すべき ATR だったとき true を返すように実装されている。

#### 4.1.3.2. 標準実装関数

OpenSC 低レベルライブラリは、多くのカードエッジインタフェース・モジュールを持っているが、ほとんどは iso7816.c にある標準実装関数を使用されている。ここでは標準実装関数が実装しているカードコマンドのうち、2.4 節で解説したコマンドについ

て実装状況を説明する。

表 34 標準実装関数とカードコマンドの対応

コマンド種別、コマンド名		参照標準	コマンドが実装されている関数
選択コマンド			
	SELECT	ISO/IEC	iso7816_select_file
	MANAGE CHANNEL	7816-4	
データ操作コマンド			
	READ BINARY	ISO/IEC	iso7816_read_binary
	WRITE BINARY	7816-4	iso7816_write_binary
	UPDATE BINARY		iso7816_update_binary
	SEARCH BINARY		
	ERASE BINARY		構造体の定義のみ
	READ RECORD		iso7816_read_record
	WRITE RECORD		iso7816_write_record
	UPDATE RECORD		iso7816_update_record
	APPEND RECORD		iso7816_append_record
	SEARCH RECORD		
	ERASE RECORD		構造体の定義のみ
	GET DATA		構造体の定義のみ
	PUT DATA		構造体の定義のみ
認証関連コマンド			
	INTERNAL AUTHENTICATE	ISO/IEC	
	GET CHALLENGE	7816-4	iso7816_get_challenge
	EXTERNAL AUTHENTICATE		
	GENERAL AUTHENTICATE		
	VERIFY		構造体の定義のみ ただし、iso7816_pin_cmd で使用
	CHANGE REFERENCE DATA		構造体の定義のみ ただし、iso7816_pin_cmd で使用
	ENABLE VERIFICATION REQUIREMENT		

	DISABLE VERIFICATION REQUIREMENT		
	RESET RETRY COUNTER		構造体の定義のみ ただし、iso7816_pin_cmd で使用
伝送処理関数			
	GET RESPONSE	ISO/IEC	iso7816_get_response
	ENVELOPE	7816-4	
MANAGE SECURITY ENVIRONMENT		ISO/IEC	
	SET	7816-4	iso7816_set_security_env
	STORE		
	RESTORE		iso7816_restore_security_env
	ERASE		
PERFORM SECURITY OPERATION		ISO/IEC	
	COMPUTE CRYPTOGRAPHIC CHECKSUM	7816-8	
	COMPUTE DIGITAL SIGNATURE		iso7816_compute_signature
	HASH		
	VERIFY CRYPTOGRAPHIC CHECKSUM		
	VERIFY DIGITAL SIGNATURE		
	VERIFY CERTIFICATE		
	ENCIPHER		
	DECIPHER		iso7816_decipher
GENERATE ASYMMETRIC KEY PAIR			

## SELECT コマンド

iso7816\_select\_file 関数で SELECT コマンドを実装している。ファイル識別子、DF 名、パスによる選択をサポートしている。選択の結果として返されるデータにはアクセス制御等に関するファイル管理情報が含まれており、iso7816\_process\_fci 関数にて処理

される。

## データ操作コマンド

標準実装ではたくさんあるデータ操作コマンドのうち、いくつか未実装なコマンドもある。未実装コマンドでも、オペレーション構造体に定義だけは存在しており、実装を各種カードエッジインターフェース・モジュールに任せようになっている。

## 認証用コマンド

標準実装関数では、INTERNAL AUTHENTICATE, EXTERNAL AUTHENTICATE, GENERAL AUTHENTICATE のいずれも単体の関数としては実装しておらず、これらはオペレーション構造体の定義もない。

## VERIFY, CHANGE REFERENCE DATA, RESET RETRY COUNTER コマンド

標準実装関数では実装されていない。オペレーション構造体の定義はある。

単体では実装されていないが、PIN のベリファイ、変更、ブロック解除を行う iso7816\_pin\_cmd 関数内部では使用している。

## MANAGE SECURITY ENVIRONMENT コマンド

MANAGE SECURITY ENVIRONMENT コマンドを単独で実行するための関数は用意されておらず、SET、STORE、RESTORE、ERASE オペレーションのうち、SET オペレーションを実装した iso7816\_set\_security\_env 関数、RESTORE オペレーションを実装した iso7816\_restore\_security\_env 関数が実装されている。STORE、ERASE オペレーションは実装されておらず、また、オペレーション構造体の定義もない。STORE ERASE オペレーションについては、現在のところ OpenSC が用意した機能では使用する必要がない。

## PERFORM SECURITY OPERATION コマンド

PERFORM SECURITY OPERATION コマンドをそのまま実行するための関数は用意されておらず、COMPUTE DIGITAL SIGNATURE オペレーションを実行するための iso7816\_compute\_signature 関数、DECIPHER オペレーションを実行するための iso7816\_dechiper 関数が実装されている。この2種類のカードコマンドはカード内部のプライベート鍵を使用するオペレーションであり、IC・ID カードが提供する機能のうち最も重要なものであるといえる。他の6種類のオペレーションはカード内のプライベ

ート鍵を使用しないため、ミドルウェアやクライアントアプリケーション側でも実装可能であり、かならずしも IC カード自身が行わねばならない機能とは言えない。OpenSC 実装ではオペレーション構造体の定義もない。

#### 4.1.3.3. MANAGE SECURITY ENVIRONMENT と PERFORM SECURITY OPERATION の関連

内部で PERFORM SECURITY OPERATION カードコマンドを実行する、iso7816\_compute\_signature 関数、iso7816\_dechiper 関数に先立って MANAGE SECURITY ENVIRONMENT カードコマンドを実行する iso7816\_set\_security\_env 関数が実行される。

例として、署名操作をする場合には、カードコマンドは次の順序で実行されることになる。

1. VERIFY  
署名に使用するプライベート鍵へのアクセス権を獲得
2. SELECT  
プライベート鍵を選択
3. MANAGE SECURITY ENVIROMENT(SET)  
今の状態が、署名動作の環境であることを宣言
4. PERFORM SECURITY OPERATION(COMPUTE DIGITAL SIGNATURE)  
現在の状態、つまり、選択されているプライベート鍵を使って署名

上記の手順が行われるとき、PKCS#11 ライブラリ、OpenSC 独自の PKCS#15 ライブラリ、OpenSC 低レベルライブラリ、カードコマンドの各レベルでの動作を表にすると次のようになる。詳細については 4.1.7 小節で解説する。

表 35 署名時の関数呼び出しの関連

PKCS#11	OpenSC PKCS#15	OpenSC 低レベルライブラリ	カードコマンド
C_Login	sc_pkcs15_verify_pin	sc_pin_cmd	VERIFY
C_Sign	sc_pkcs15_compute_signature	sc_select_file	SELECT

C_SignFinal	sc_set_security_env	MANAGE SECURITY ENVIRONMENT
	sc_compute_signature	PERFORM SECURITY OPERATION

C\_Sign と C\_SignFinal は署名生成部分について、同じ処理を行っている。

#### 4.1.4. データモデルとの対応付け

OpenSC 低レベルライブラリが、カード内のファイルを操作するレベルの機能を提供していたのに対し、OpenSC PKCS#15 ライブラリでは、より高次の操作を提供する。

OpenSC PKCS#15 ライブラリのレイヤーが、カードに格納されているデータのフォーマット、データモデルに依存する操作を行う際の基本的な操作順序は次のようになる。

1. OpenSC PKCS#15 ライブラリが、OpenSC 低レベルライブラリを使用し、カードからデータを読み込んで OpenSC が定義する PKCS#15 構造体に保存
2. OpenSC PKCS#15 ライブラリが PKCS#15 構造体を参照
3. OpenSC PKCS#15 ライブラリが、OpenSC 低レベルライブラリを使用し、カードに署名操作させる

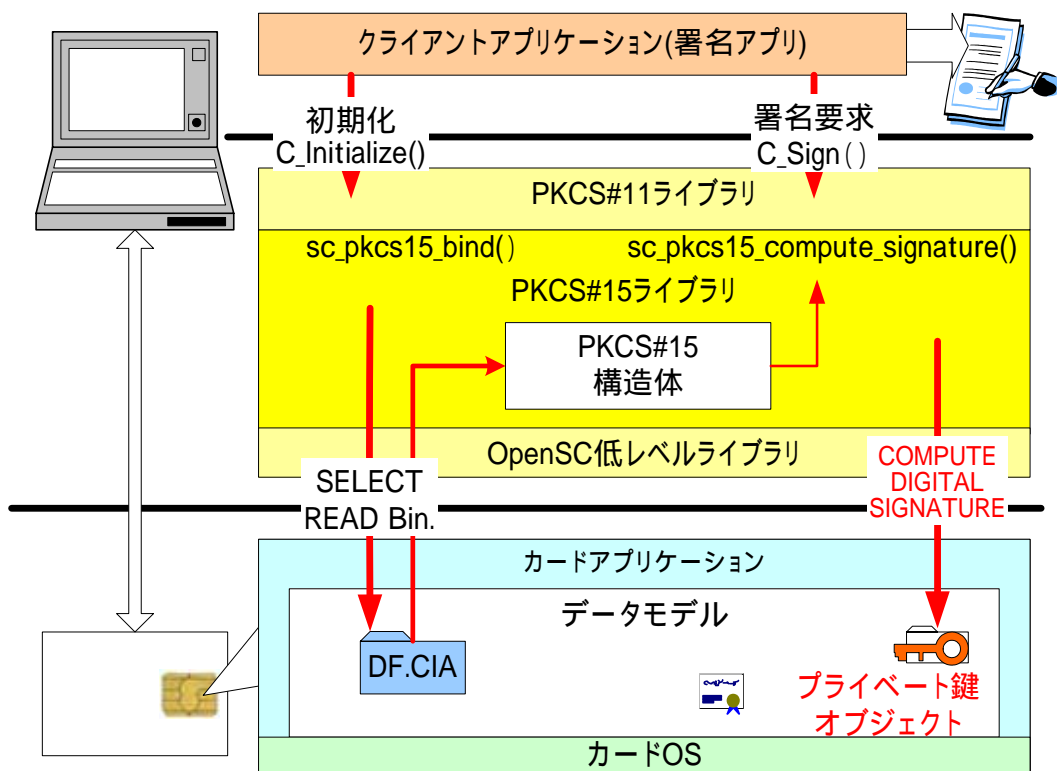


図 32 データモデルと PKCS#15 構造体のマップ

OpenSC PKCS#15 ライブラリはカードの中身を必要となった時点で直接参照するのではなく、OpenSC が独自に定義した PKCS#15 構造体データを参照して動作を行う。そのためさまざまな操作を行う前にカードに保存されているデータを PKCS#15 構造体に対応させる必要がある。この操作はカードに保存されているデータモデルに依存しており、標準実装では FINEID や BELPIC のような PKCS#15 準拠カードであることを前提としている。準拠カードでない場合 ID カードのデータモデルに依存してデータの読み出しを行う必要があるため、ID カードごとに、データモデルがハードコードされたエミュレーション用のモジュールを用意している。

対象としている ID カードが PKCS#15 準拠である場合には、カードからデータを読み込み、PKCS#15 構造体に格納する部分には OpenSC PKCS#15 ライブラリの標準関数を使用する (図 33)。

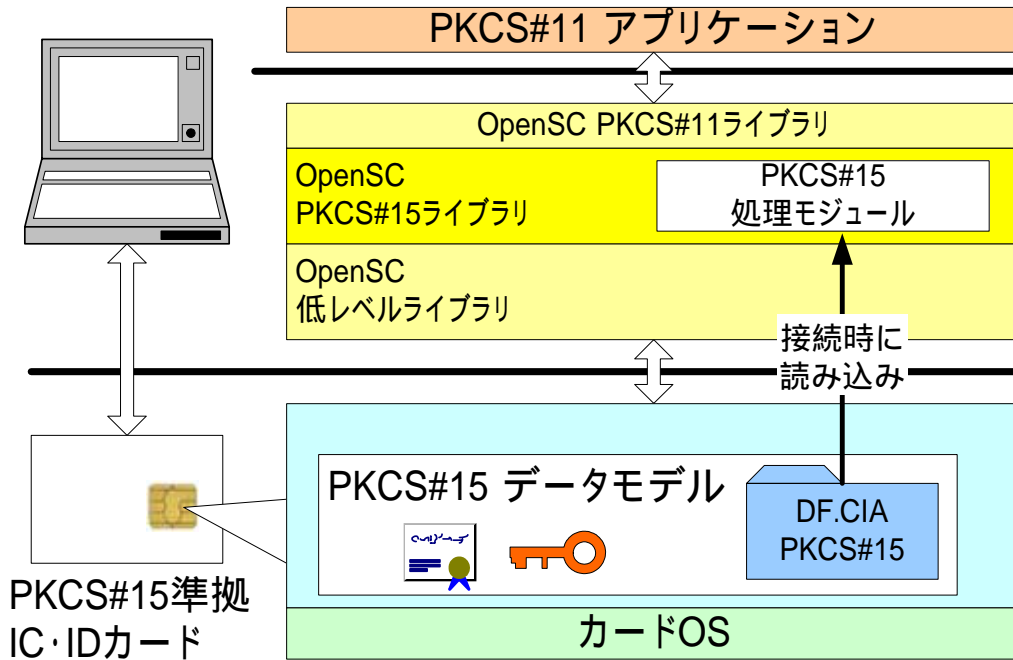


図 33 PKCS#15 準拠データモデルの読み込み

カードが PKCS#15 準拠でない場合、データを読み込んで PKCS#15 構造体にマップする作業を対象カード専用のエミュレーションモジュールが実行する。PIV II は PKCS#15 準拠ではないため、PKCS#15 構造体にデータを格納する部分を独自に実装している [pkcs15-piv.c]。なお、PIV ではデータをマップする PKCS#15 エミュレーションモジュールだけでなく、カードエッジインターフェースにも専用のカードエッジインタフェース・モジュールを使用している (図 34)。

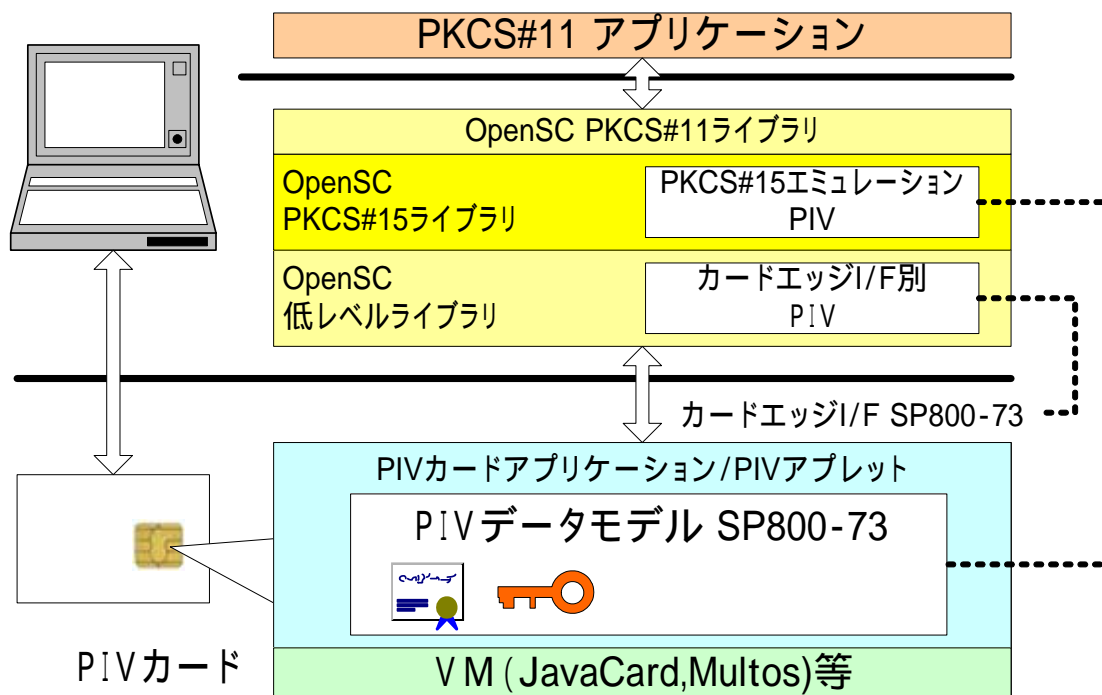


図 34 PIV データモデルの読み込み

一度 PKCS#15 構造体に格納された後は、共通関数でそのデータを参照するため、IC カードに直接アクセスする必要がない操作に関しては、IC・ID カードごとの差異は生じない。PKCS#15 構造体中のデータを IC カードに書き込む、または IC カードに計算させる作業が発生した場合、IC・ID カードごとの違いについては OpenSC 低レベルライブラリが吸収する。なお、OpenSC PKCS#15 ライブラリとは OpenSC が独自に定義、実装したライブラリであり、なんら標準化された仕様ではない点に注意が必要である。

PKCS#11 ライブラリから呼ばれる OpenSC PKCS#15 ライブラリ関数には表 36 OpenSC PKCS#15 ライブラリの関数のようなものがある。

表 36 OpenSC PKCS#15 ライブラリの関数

関数名	説明
sc_pkcs15_bind	IC/ID カードからデータを読み込み、PKCS#15 構造体にマップする。
sc_pkcs15_compute_signature	署名の生成を行う。
sc_pkcs15_decipher	暗号データの複合を行う。
sc_pkcs15_verify_pin	PIN の認証を行う

unbind	
--------	--

#### 4.1.4.1. PKCS#15 準拠データモデル読み込みに関する実装

OpenSC の標準実装では、IC・ID カードのデータモデルが PKCS#15 に準拠していることを前提として、データの読み込みを行う。読み込んだデータは、PKCS#15 構造体にマップされる。OpenSC 独自データ構造を以下に示す。

```
typedef struct sc_pkcs15_card {
    sc_card_t *card;
    char *label;
    /* fields from TokenInfo: */
    int version;
    char *serial_number, *manufacturer_id;
    char *last_update;
    unsigned int flags;
    struct sc_pkcs15_algorithm_info alg_info[1];

    sc_file_t *file_app;
    sc_file_t *file_tokeninfo, *file_odf, *file_unusedspace;

    struct sc_pkcs15_df *df_list;
    struct sc_pkcs15_object *obj_list;
    int record_lengths[SC_PKCS15_DF_TYPE_COUNT];
    sc_pkcs15_unusedspace_t *unusedspace_list;
    int unusedspace_read;

    struct sc_pkcs15_card_opts {
        int use_cache;
    } opts;

    unsigned int magic;

    void *dll_handle; /* shared lib for emulated cards */
    char *preferred_language;
} sc_pkcs15_card_t;
```

PKCS#15 データが読み込まれると、obj\_list に CIO (Cryptographic information object) が格納される。以後の操作では、基本的にはこの obj\_list を扱う。

代表的なデータについて読み込み部分の実装を説明する。

表 37 データモデルと読み込み関数

対象オブジェクト	読み込み実装関数	対象データの内容
EF.DIR カードアプリケーション情報	sc_enum_apps	アプリケーション配置を示すデータ。このデータから DF.CIA のパスを知り、DF.CIA を読み込む際に指定する。
DF.CIA 暗号情報アプリケーションディレクトリ	sc_pkcs15_bind	PKCS#15 アプリケーションディレクトリ。
EF.CIAInfo 暗号情報アプリケーション情報	sc_pkcs15_parse_tokeninfo	CIA の情報やカード自体の情報など。このデータから得られたデータは PKCS#11 ライブラリのレベルで UI に使用される。
EF.OD オブジェクトディレクトリ	parse_odf	暗号情報オブジェクトのリスト。このデータから EF.AOD,EF.PrKD など、他のオブジェクトのパスなどを知る。
EF.AOD 認証情報オブジェクト	sc_pkcs15_decode_aodf_entry	認証方法に関する暗号情報オブジェクト。このオブジェクトを指定して VERIFY カードコマンドを実行することで、このオブジェクトに関連付けられているプライベート鍵等にアクセスできるようになる。
EF.PrKD プライベート鍵情報オブジェクト	sc_pkcs15_decode_prkdf_entry	プライベート鍵に関する暗号情報オブジェクト。このオブジェクトからプライベート鍵の実体ファイルのパスを知ることができる。
EF.PuKD 公開鍵情報オブジェクト	sc_pkcs15_decode_pukdf_entry	公開鍵に関する暗号情報オブジェクト。このファイルから公開鍵の実体ファイルへのパスを知ることができる。
EF.CD X509 証明書情報オブジェクト	sc_pkcs15_decode_cdf_entry	証明書に関する暗号情報オブジェクト。このファイルから証明書の実体ファイルへのパスを知ることができる。

## EF.DIR

OpenSC PKCS#15 ライブラリでは、sc\_pkcs15\_bind 関数[pkcs15.c]を使ってカードからデータを読み込む。その最初の段階で sc\_enum\_apps 関数[dir.c]をコールし、EF.DIR の読み込みを行う。EF.DIR をパースし、OpenSC 独自データ構造のアプリケーション領域にデータをセットする。複数レコードにも対応している。

## DF.CIA

次に、DF.CIA の読み込みを行う。先に読んだ EF.DIR を使って DF.CIA を検索する。その際、sc\_find\_pkcs15\_app 関数[dir.c]を使用するが、ここでは EF.DIR に書かれた AID(Application ID)が次のどちらかにマッチするアプリケーションを探している。CIA が複数あっても、選択する機構は準備されていない。

表 38 OpenSC が読み込む AID

値	備考
A0 00 00 00 63 50 4B 43 53 2B 31 35	PKCS #15 ( FINEID など )
A0 00 00 01 77 50 4B 43 53 2B 31 35	BELPIC

## EF.CIAInfo

EF.CIAInfo をパースし、sc\_pkcs15\_tokeninfo 構造体につめる。CIAInfo の持っているデータのうち、いくつかは読み込まず破棄している。読み込んだデータは PKCS#11 ライブラリから参照可能である。

表 39 OpenSC が読み込む CIAInfo のデータ

データ	読み込んでいるデータの例(4.1.7 節を参照)
version	1
serialNumber	0000DD0AFFFFF0200
manufactureID	OpenSC Project
label	CardHolder
cardflags	0x3
seInfo	読み込んでいない
recordInfo	読み込んでいない

supportedAlgorithms	読み込んでいない
issureId	読み込んでいない
holderId	読み込んでいない
lastUpdate	20061120072405Z
preferredLanguage	en
profileIndication	読み込んでいない

### EF.OD (オブジェクトディレクトリ)

EF.OD をパースして PKCS#15 構造体につめる。

### EF.AOD (認証情報オブジェクト)

EF.AOD に格納されているデータは、OpenSC では `sc_pkcs15_decode_aodf_entry[pkcs15-pin.c]` によってパースされ、`sc_pkcs15_pin_info` 構造体にマップされる。

```
struct sc_pkcs15_pin_info {
    struct sc_pkcs15_id auth_id;
    int reference;
    unsigned int flags, type;
    size_t min_length, stored_length, max_length;
    u8 pad_char;
    struct sc_path path;
    int tries_left;

    unsigned int magic;
};
```

EF.PrKD にアクセスするためには、EF.PrKD の `auth_id` と一致する `auth_id` (AOD の構造体メンバー) を持つ AOD を指定して VERIFY カードコマンドに成功していなければならない。

読み込まれた属性の `PasswordFlag` は `info.flag` に格納されるが、ほとんどの値は OpenSC PKCS#15 ライブラリや OpenSC 低レベルライブラリでは参照されず、PKCS#11 ライブラリから参照される。`needs-padding` は PIN コードを送るとき、未使用バッファを `0xFF` などでパディングするか否かの判断に使用している。

### EF.PrKD (プライベート鍵情報オブジェクト)

EF.PrKD に格納されるデータは OpenSC では `sc_pkcs15_decode_prkdf_entry[pkcs15-prkey.c]` によってパースされ、`sc_pkcs15_prkey_info` 構造体にマップされる。

```

struct sc_pkcs15_prkey_info {
    struct sc_pkcs15_id id; /* correlates to public certificate id */
    unsigned int usage, access_flags;
    int native, key_reference;
    size_t modulus_length;
    u8 *subject;
    size_t subject_len;

    struct sc_path path;
};

```

対応しているデータは RSA プライベート鍵、DSA プライベート鍵の 2 種類で、楕円曲線プライベート鍵、Diffie-Hellman プライベート鍵、KEA プライベート鍵、汎用プライベート鍵を読み込むことはできない。EF.PrDK に限らず、EF.OD はさまざまな属性を持っている。その属性には、すべての OD が持つ属性である CommonObjectAttributes、プライベート鍵、公開鍵などのすべての鍵オブジェクトがもつ CommonKeyAttributes、RSA プライベート鍵のみが持つ PrivateRSAKeyAttributes などがある。OpenSC は EF.PrKD からこれらの属性を読み込み、sc\_pkcs15\_prkey\_info 構造体にセットするが、最終的にはすべての情報ではなく、いくつかのデータのみがマップされ、使用される。

表 40 sc\_pkcs15\_prkey\_info とカード内データモデルの対応

構造体メンバ	カード内データモデル
usage	CommonKeyAttributes.usage
access_flag	CommonKeyAttributes.accessFlags
native	CommonKeyAttributes.native
key_reference	CommonKeyAttributes.keyReference
modules_length	PrivateRSAKeyAttributes.moduleLength
subject	未設定
subject_len	未設定
path	PrivateRSAKeyAttribute.value PrivateDSAKeyAttribute.value

プライベート鍵を扱う際は、表 40 にあるデータだけでなく、CommonObjectAttributes の値が重要である。CommonObjectAttributes は、プライベート鍵のみが持っている属性ではなく、一般にすべての CIO が持つ属性であり、

OpenSC 実装でも、すべての CIO を読み込む際、sc\_pkcs15\_object 構造体(pkcs15.h) に一律に読み込まれる。

表 41 sc\_pkcs15\_object とカード内データモデルの対応

構造体メンバ	カード内データモデル
label	CommonObjectAttributes.label
flags	CommonObjectAttributes.flags
auth_id	CommonObjectAttributes.authId
user_consent	CommonObjectAttributes.userConsent
対応なし	CommonObjectAttributes.accessControlRules

ここの auth\_id と同じ auth\_id をもつ AOD が、このプライベート鍵を使用する場合に VERIFY しておかなければならない AOD である。

### EF.PuKD (公開鍵)

sc\_pkcs15\_decode\_pubkf\_entry 関数[pkcs15-pubkey.c]を使ってカードからデータを読み込み、sc\_pkcs15\_pubkey\_info 構造体にマップされる。

```
struct sc_pkcs15_pubkey_info {
    struct sc_pkcs15_id id; /* correlates to private key id */
    unsigned int usage, access_flags;
    int native, key_reference;
    size_t modulus_length;
    u8 *subject;
    size_t subject_len;

    struct sc_path path;
};
```

読み込めるデータは RSA 公開鍵、DSA 公開鍵の 2 種類に対応している。

### EF.CD (証明書)

sc\_pkcs15\_decode\_cdf\_entry 関数によって sc\_pkcs15\_cert\_info 構造体にマップされる。

```
struct sc_pkcs15_cert_info {
    struct sc_pkcs15_id id; /* correlates to private key id */
    int authority; /* boolean */
    /* identifiers [2] SEQUENCE OF CredentialIdentifier{{KeyIdentifiers}} */
    struct sc_path path;
};
```

```

sc_pkcs15_der_t value;
};

```

読み込めるのは X.509 証明書と X.509 属性証明書のみで、他の型には対応していない。

表 42 EF.CD のデータと構造体メンバの対応

構造体メンバ	データモデル
id	CommonCertificateAttribute.iD
authority	CommonCertificateAttribute.authority
path	CertAttribites.value
value	CertAttributes.value

#### 4.1.5. プライベート鍵の操作と API

ISO/IEC7816-8 に規定されるセキュリティオペレーションは 8 種類あるが、4.1.3.3 節でも述べたとおり OpenSC 低レベルライブラリで実装されるオペレーションは COMPUTE DIGITAL SIGNATUR と DECIPHER の 2 種類のみである。OpenSC PKCS#15 ライブラリ中の `sc_pkcs15_compute_signature` 関数と `sc_pkcs15_decipher` 関数からそれぞれ OpenSC 低レベルライブラリの関数を呼び出し、カードコマンドを実行している。

どちらの関数もプライベート鍵への参照を引数で受け取り、内部で `sc_select_file` 関数にて選択操作を行うため、プライベート鍵と結びついている (`auth_id` が一致する) AOD の認証 (VERIFY) を行っておかなければならない。

#### `sc_pkcs15_compute_signature` 関数

この関数は署名を行う関数であるが、署名に使用するプライベート鍵が認証用であれば認証動作であり、否認防止用であれば、いわゆる、否認防止用の署名動作になる。この関数は引数として、署名に使用するプライベート鍵への参照を受け取る。はじめにそのプライベート鍵の `native` 属性をチェックし、`false` だった場合エラーコード `SC_ERROR_EXTRACTABLE_KEY` を返す。プライベート鍵の `native` 属性は、ISO/IEC7816-15 によると、「鍵に関連する暗号アルゴリズムがカードハードウェアに実装されているか否か」を表すフラグである。IC・ID カードでは、`native` 属性が `true` と

なるべきである。

sc\_select\_file 関数でプライベート鍵を選択、sc\_set\_security\_env 関数をつかって環境をセットし、sc\_compute\_signature 関数を呼び出して、最終的に署名を行う。使用する鍵がわかるのは sc\_set\_security\_env までであり、sc\_compute\_signature 関数の内部ではどの鍵を使用するのか判別できない。

### sc\_pkcs15\_decipher 関数

sc\_pkcs15\_compute\_signature 関数と同様、復号に使用するプライベート鍵への参照を引数で受け取る。はじめにプライベート鍵の native 属性をチェックし、false だった場合エラーコード SC\_ERROR\_EXTRACTABLE\_KEY を返す。これも同様に、sc\_select\_file 関数でプライベート鍵を選択、sc\_set\_security\_env 関数で環境をセットした後 sc\_pkcs15\_decipher 関数をコールして、復号を行う。

## 4.1.6. 各種 IC・ID カードの対応

### 4.1.6.1. FINEID カードの実装

FINEID は ISO/IEC7816-15 準拠のデータモデルを採用したカードであり、データモデルを PKCS#15 構造体にマップする部分の特別な実装は提供されていない。カードエッジインタフェース・モジュールでは、SetCOS をサポートするカードエッジインタフェース・モジュールが FINEID をハンドリングしている。

SetCOS ([setec 社製]のカード) をサポートするための [card-setcos.c] に FINEID に関連するコードが実装されているが、FINEID が必ずしも SetCOS を必要としているわけではない。

### 対象カードか否かの判定(setcos\_match\_card)

ソースコードには 11 種類の ATR がハードコードされており、カードから返された ATR がそのうちのいずれかにマッチするか、もしくは ATR の Historical bytes に「FinEID」という文字列が含まれていた場合、このカードエッジインタフェース・モジュールの対象カードであると判断している。Historical bytes に「FinEID」があるときの動作は、FINEID-S4-1/2.1A 3.1.ATR bytes にあるサンプルデータがそのまま使用された場合を想定している。

#### 4.1.6.2. BELPIC カードの実装

BELPIC は PKCS#15 準拠のデータモデルを採用したカードである。データモデルを PKCS#15 構造体にマップする部分の特別な実装は用意されていない。BELPIC 用のカードエッジインタフェース・モジュール[card-belpic.c]が用意されており、ソースコード中のコメントには「Belpic カードは Cyberflex Java card だが、それに依存しないので、ドライバが ( card-flex.c と ) 分離されている」と書かれている。BELPIC カードエッジインタフェースを実現する、Java アプレットがあり、Cyberflex に依存しないことを表明しているものであると思われる。

card-belpic.c には利用者に PIN パッドからの入力を促す際に表示する文字列や、エラーメッセージなども実装されており、本来であればもっと高いレイヤーに実装されるべきコードも含まれている。

BELPIC 用カードエッジインタフェース・モジュールが標準実装を置き換えている部分は次のようになっている。

```
static struct sc_card_driver *sc_get_driver(void)
{
    if (iso_ops == NULL)
        iso_ops = sc_get_iso7816_driver()->ops;

    belpic_ops.match_card = belpic_match_card;
    belpic_ops.init = belpic_init;
    belpic_ops.finish = belpic_finish;

    belpic_ops.select_file = belpic_select_file;
    belpic_ops.read_binary = belpic_read_binary;
    belpic_ops.pin_cmd = belpic_pin_cmd;
    belpic_ops.set_security_env = belpic_set_security_env;
    belpic_ops.logout = belpic_logout;

    belpic_ops.compute_signature = belpic_compute_signature;
    belpic_ops.get_challenge = iso_ops->get_challenge;
    belpic_ops.get_response = iso_ops->get_response;
    belpic_ops.check_sw = iso_ops->check_sw;

    return &belpic_drv;
}
```

#### 対象カードか否かの判定(belpic\_match\_card)

ソースコードに 5 種類の ATR がハードコードされており、ATR にマッチした場合 BELPIC カードであるとしている。

#### belpic\_init

BELPIC カードは、BELPIC カードアプリケーションのほかに、ID カードアプリケーションを持っている。OpenSC は BELPIC カードの ID カードアプリケーション中にある EF.Preferences を読み込み、language 属性を使用している。EF.Preferences のパスは 3F00/DF01/4039 と決められている。このファイルを read\_binary オペレーションを使って読み込み、パースして得た「language」情報を OpenSC 独自データ構造のカード固有データ領域に格納している。

language 属性は、エラーメッセージなどを表示する際に利用されている。CIAInfo の preferredLanguage は使用されていない。

### **belpic\_set\_security\_env**

BELPIC では DECIPHER セキュリティオペレーションが規定されていないため、SET SECURITY ENVIRONMENT コマンドでも署名用オペレーションのみサポートしている。

set\_security\_env 関数で署名用にセットする鍵リファレンスの使用目的が nonRepudiation (否認防止) だったとき、GUI、または PIN パッドを使ったユーザによる PIN コードの入力を必須としている。この処理は本来の意味からすると署名関数である compute\_signature 内部で行うべきであるが、compute\_signature 関数では、いまから行う署名に使用する鍵がどの鍵なのかを知ることができないため、compute\_signature 関数に先立って実行される set\_security\_env 関数で実行している。

署名操作を行うとき、ユーザに PIN の入力を求めるべきか否かを判断する材料としては、CommonObjectAttribute.userConsent を使用するべきである。OpenSC 実装では、userConsent を読み込み、sc\_pkcs15\_object.user\_consent にマップしており、pkcs#11 ライブラリにて、PIN をキャッシュするか否かに使用されている。

### **belpic\_compute\_signature**

標準の [iso7816.c]compute\_signature をコールし、エラーコードとして「Security status not satisfied」が返された場合、GUI を使用して認証用に PIN コードを入力させ、検証してから再度 compute\_signature をコールする実装となっている。先の set\_security\_env 関数において、鍵の使用目的が nonRepudiation の時、常に PIN コードを入力させていたのとは違い、こちらは署名操作が失敗したときのみ、PIN を入力して再度署名を行う。したがって、一度正しい PIN を入力すればカードが拒否しない限り複数回の署名操作が可能な実装になっている。ただし、カード側の実装が署名のたびに VERIFY を求めるようになっている場合、ミドルウェアが VERIFY なしで署名を行お

うとすると拒否されるため、続けて署名できるのはあくまでもカードの実装が許せば、という前提がある点に注意が必要である。

## decipher

BELPIC カードエッジインタフェース・モジュールは decipher 関数をオーバーライドしておらず、標準実装が提供する iso7816\_decipher 関数がセットされたままになっている。

BELPIC では、カードコマンドとして DECIPHER が提供されていない。そのため、アプリケーションが sc\_decipher 関数をコールし、標準実装の iso7816\_decipher が実行されると、APDU がカードまで到達し、その時点でエラーとなるはずである。

### 4.1.6.3. PIV カードの実装

PIV のデータモデルは、ISO/IEC7816-15、PKCS#15 非準拠である。また、カードエッジも ISO/IEC7816-4,8 に準拠してはいるものの、3.4.3 節で述べられているように実装済みコマンドに READ BINARY がない、データモデルとして基本的に「ファイル」という概念を持たないなど、BELPIC、FINEID などと比べてユニークな仕様である。そのため、カードエッジインタフェース・モジュールと、PKCS#15 エミュレータの両方が専用に提供されている。

PKCS#15 エミュレータはその初期化関数の中で、3.4.3 節で説明したデータオブジェクト、各種証明書、PIN、プライベート鍵、公開鍵などのすべての情報をハードコードしたデータを使って、PKCS#15 構造体を初期化している。このとき、カードからはどのようなデータも取得しておらず、後で実際にカード内のデータが必要になったとき、取得するような実装がなされている。

PKCS#15 構造体には、たとえばパス DB00 に” Card Capability Container”というラベルがついたファイルが存在しているようなデータをセットし、後述する sc\_select\_file 関数によって DB00 が選択され、sc\_read\_binary 関数によってデータを読み込むという操作が行われたとき、その動作をエミュレート可能とするための準備をする。

PIV 用カードエッジインタフェース・モジュールは PIV エンドポイントインタフェースを対象としており、標準実装を置き換えている部分は次のようになっている。

```
static struct sc_card_driver * sc_get_driver(void)
{
    struct sc_card_driver *iso_drv = sc_get_iso7816_driver();

    piv_ops = *iso_drv->ops;
    piv_ops.match_card = piv_match_card;
}
```

```

piv_ops.init = piv_init;
piv_ops.finish = piv_finish;

piv_ops.select_file = piv_select_file; /* must use get/put, could emulate? */
piv_ops.logout = piv_logout;
piv_ops.get_challenge = piv_get_challenge;
piv_ops.read_binary = piv_read_binary;
piv_ops.write_binary = piv_write_binary;
piv_ops.set_security_env = piv_set_security_env;
piv_ops.restore_security_env = piv_restore_security_env;
piv_ops.compute_signature = piv_compute_signature;
piv_ops.decipher = piv_decipher;
piv_ops.card_ctl = piv_card_ctl;

return &piv_drv;
}

```

### **piv\_match\_card**

常に false を返すように実装されている。つまり、PIV ドライバでは ATR によってそのカードが PIV であるかどうかを判断しない。

### **piv\_select\_file**

PIV カードエッジインタフェース・モジュールはデータオブジェクトのコンテナ ID、BER-TLV タグ、データタイプなどを PIV データオブジェクトテーブルとして持っている。

PIV の SELECT コマンドでは、AID を指定して DF を選択する機能しか実装されないため、パスを指定してファイルを選択するというオペレーションは、そのままでは実装できない。PIV エミュレータの実装では、piv\_select\_file で指定されたパスをコンテナ ID として扱い、PIV データオブジェクトテーブルから検索し、レコードを PIV エミュレータのプライベート領域に記憶する実装になっている。ここで記憶した PIV データオブジェクトレコードから取得した BER-TLV タグを、後で piv\_read\_binary , piv\_write\_binary 関数内で GET DATA カードコマンド、PUT DATA カードコマンドのパラメータとして使用し、データオブジェクトを取得・書き込みしている。

### **piv\_get\_challenge**

PIV カードは GET CHALLENGE カードコマンド[ISO/IEC7816-4 7.5.3]を持たない。かわりに、GENERAL AUTHENTICATE カードコマンドの動的認証テンプレート「質問」(タグ = 0x81) を利用して、チャレンジコードを取得している。

## **piv\_read\_binary**

先に実行される `piv_select_file` で得られた BER-TLV タグを、GET DATA のパラメータとして使用し、データを取得する。`read_binary` コマンドを呼び出す側は、タグのない形式のデータを期待しているはずだが、PIV の GET DATA コマンドは、タグ付きのデータを返す。`piv_read_binary` はタグを取り除く。

## **piv\_write\_binary**

先に実行される `piv_select` で、データを書き込む対象となるデータオブジェクトは確定している。対象が証明書 ( PIV 認証に対する X.509 証明書、否認防止用署名に対する X.509 証明書、鍵管理に対する X.509 証明書、カード認証に対する X.509 証明書 ) でない場合、エラー `SC_ERROR_NOT_SUPPORTED` を返す。

## **セキュリティオペレーション**

`piv_set_security_env`、`piv_restore_security_env` コマンドに対応するカードコマンドはない。PIV カードエッジインタフェース・モジュール実装では、`piv_set_security_env` に引数として渡される鍵への参照情報を、PIV エミュレータのプライベート領域に保存する。`piv_restore_security_env` 実装の中身は空である。

署名生成、復号処理のためには `piv_set_security_env` のあと、`piv_compute_signature` や `piv_decipher` コマンドがコールされるが、どちらの場合も `piv_validate_general_authentication` 関数を呼び出している。

```
static int piv_compute_signature(sc_card_t *card,
                               const u8 * data, size_t datalen,
                               u8 * out, size_t outlen)
{
    SC_FUNC_CALLED(card->ctx, 4);
    return piv_validate_general_authentication(card, data, datalen, out, outlen);
}

static int piv_decipher(sc_card_t *card,
                       const u8 * data, size_t datalen,
                       u8 * out, size_t outlen)
{
    SC_FUNC_CALLED(card->ctx, 4);

    return piv_validate_general_authentication(card, data, datalen, out, outlen);
}
```

`piv_validate_general_authentication` 関数内部では GENERAL AUTHENTICATE カードコマンドを実行している。カードコマンドの名前には AUTHENTICATE とついているが、ここでは署名、復号オペレーションに使用している。

署名も、復号もまったく同じ引数で `piv_validate_general_authenticate` を呼び出していることから、この関数の前に実行された `set_security_env` でセットされたデータ、つまり、鍵への参照が GENERAL AUTHENTICATE の動作を決定していることがわかる。

### 認証オペレーション

PIV カードの実装では、GENERAL AUTHENTICATE カードコマンドを使用した外部認証関数 `piv_general_external_authenticate`、相互認証関数 `piv_general_mutual_authenticate` が実装されている。

GENERAL AUTHENTICATE カードコマンドは与える引数によって次の機能を実行する。

1. nonce の生成と、生成した nonce の記憶
2. 暗号化
3. 復号
4. 復号したデータと、記憶している nonce の比較による認証
5. 暗号化済み nonce の生成と記憶

`piv_general_external_authenticate` 関数が行う nonce 要求と、検証要求が GENERAL AUTHENTICATE カードコマンドである。

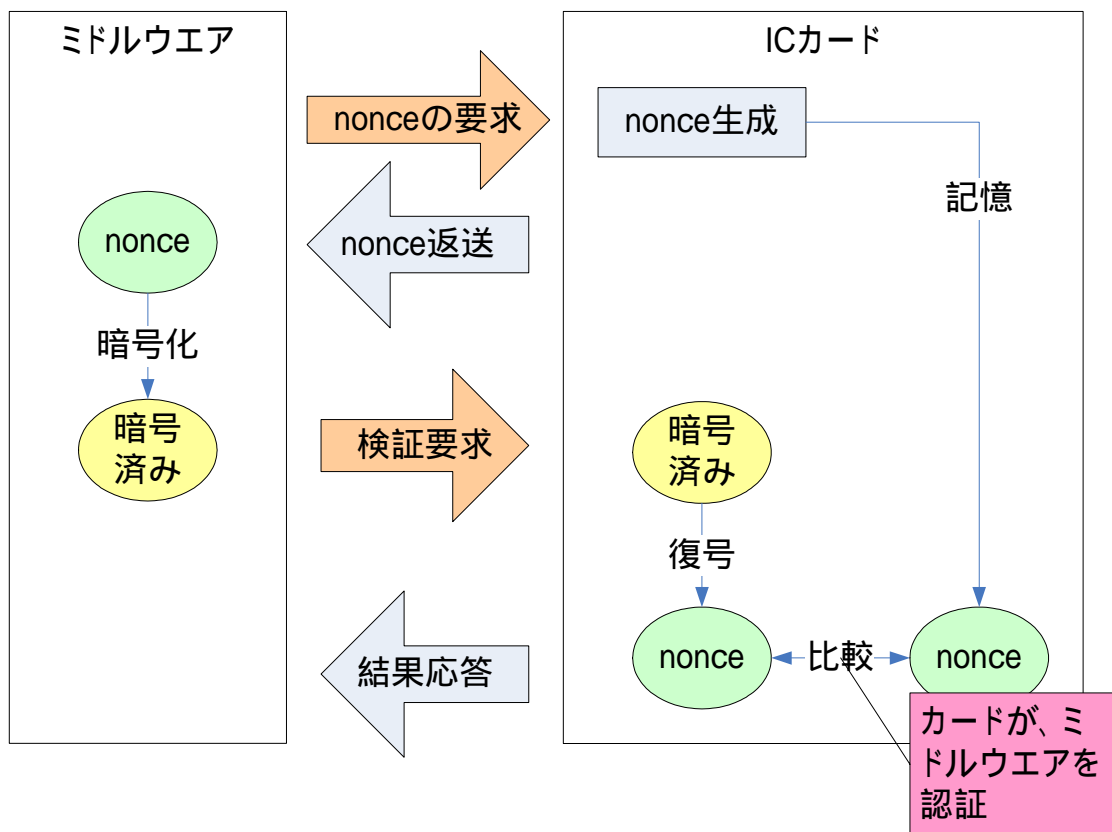


図 35 `externa_authenticate` の流れ

`piv_general_mutual_authenticate` 関数では暗号化済み `nonceA` の要求、復号済み `nonceA` と新しい `nonceB` の送信、検証要求部分が `GENERAL AUTHENTICATE` である。

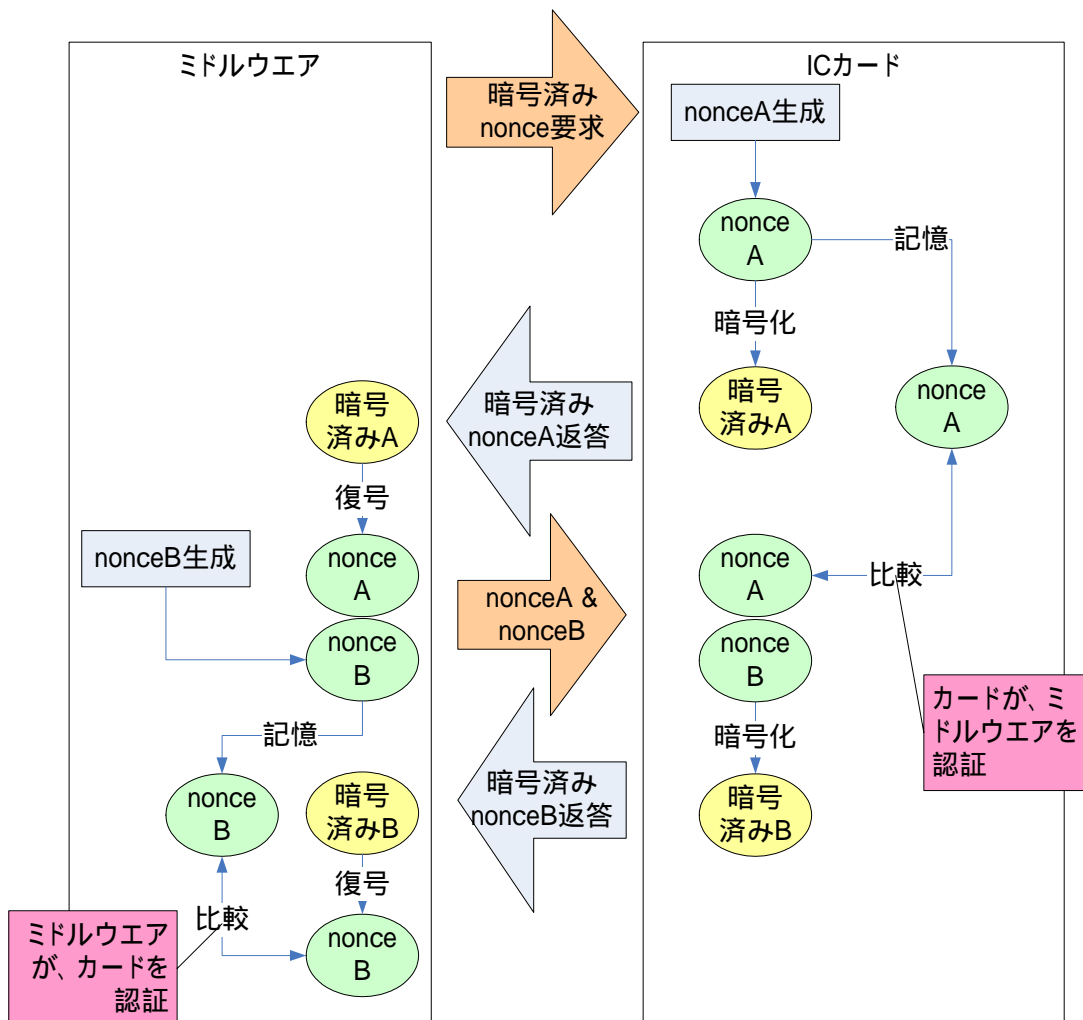


図 36 mutual\_authenticate の流れ

#### 4.1.7. 署名操作の流れ

PKCS#11API のレイヤーからみて、任意のファイルに IC カードを使用して署名操作を行うときの流れを簡単にまとめる。実際に pkcs11-tool コマンドラインツールで署名生成をするときの動作をトレースし、PKCS#11 からカードコマンドにいたる流れを説明する。

カードは Axalto 社の Cryptoflex 32K、カードリーダーは Reflex USB v3 を使用した。pkcs15-init コマンドラインツールを使用して PKCS#15 フォーマットで初期化し、プライベート鍵、それに対応する公開鍵証明書と、ルート証明書、中間証明書をインストールしたものを使用した。カードの内容を pkcs15-tool で表示すると次のようになる。

```
PKCS#15 Card [OpenSC Card]:
  Version      : 1
  Serial number : 0000DD0AFFFFF0200
  Manufacturer ID: OpenSC Project
  Last update  : 20061120072405Z
  Flags        : EID compliant

PIN [Security Officer PIN]
  Com. Flags: 0x3
  ID        : ff
  Flags     : [0xB2], local, initialized, needs-padding, soPin
  Length    : min_len:6, max_len:8, stored_len:8
  Pad char  : 0x00
  Reference : 2
  Type      : ascii-numeric
  Path      : 3f005015

PIN [CardHolder]
  Com. Flags: 0x3
  ID        : 01
  Flags     : [0x32], local, initialized, needs-padding
  Length    : min_len:4, max_len:8, stored_len:8
  Pad char  : 0x00
  Reference : 1
  Type      : ascii-numeric
  Path      : 3f0050154b01

Private RSA Key [Private Key]
  Com. Flags : 3
  Usage      : [0x10E], decrypt, sign, signRecover, derive
  Access Flags: [0x1D], sensitive, alwaysSensitive, neverExtract, local
  ModLength  : 1024
  Key ref    : 0
  Native     : yes
```

セキュリティオブジェクト

対応している。  
PrivateKey にアクセスするには PIN の認証が必要

```

Path      : 3f0050154b0130450012
Auth ID   : 01 ←
ID        : 45 ←

X.509 Certificate [/C=JP/ST=Tokyo/O=ROBOC/CN=Who am I/emailAddress=info@roboc.com
]
Flags    : 2
Authority: no
Path     : 3f0050154545
ID       : 45

X.509 Certificate [/C=JP/ST=Tokyo/L=Nerima-ku/O=ROBOC/CN=ROBOC root CA]
Flags    : 2
Authority: yes
Path     : 3f0050154546
ID       : 46

X.509 Certificate [/C=JP/ST=Tokyo/O=ROBOC/CN=ROBOC mid CA]
Flags    : 2
Authority: yes
Path     : 3f0050154547
ID       : 47

```

対応している

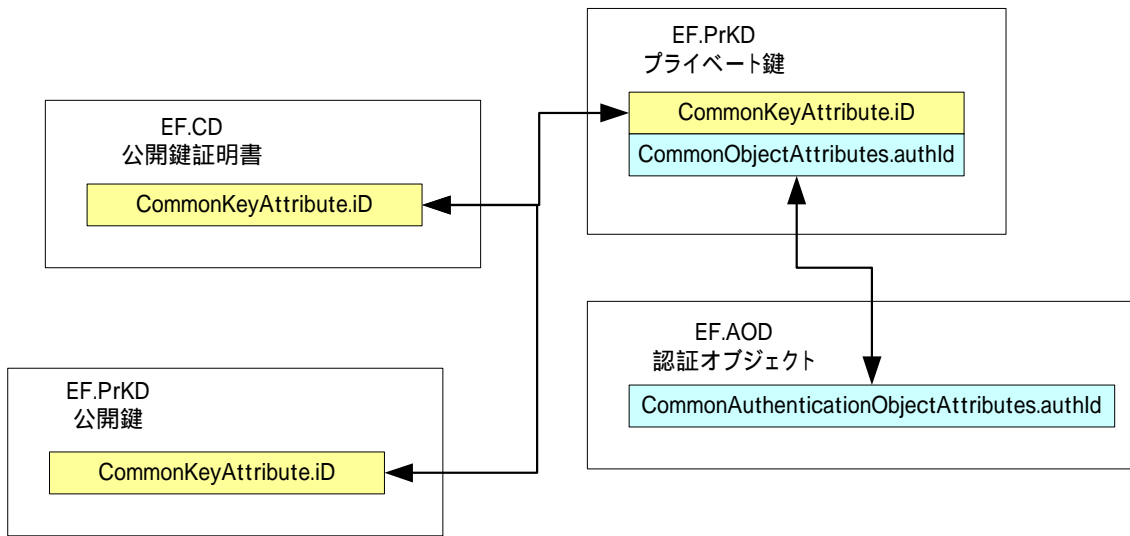


図 37 証明書、鍵、PIN の関連

pkcs11-tool で署名を行う際のコマンドラインは次のようになる。

```

$ pkcs11-tool --module /usr/local/lib/opensc-pkcs11.so --sign --output-file data.s
ign ¥
--input-file data
$ Please enter User PIN:
$ Using signature algorithm RSA-X-509
$

```

このコマンドを実行すると data に署名をし、data.sign ファイルが生成される。

元データ

```

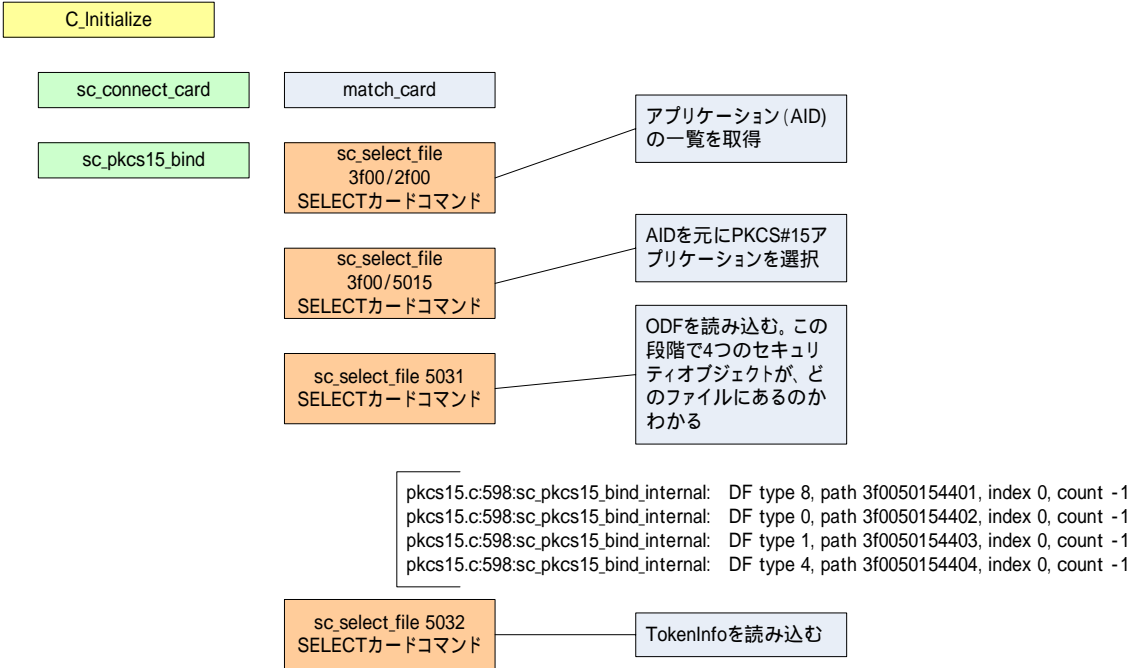
$ cat data
This is TEST.
  正常時
$ openssl rsautl -verify -in signed.data -certin -inkey ../../cert/certificate.pem
-raw
This is TEST.
$
  改ざん時
$ openssl rsautl -verify -in signed.data -certin -inkey ../../cert/certificate.pem
-raw
) ?M?5?) ?`??P?WA=
    ¥j????????9????5?L??{?eLx?053) □?@? ?r^Fw???4bUbv??d?+?Y????{?FRpz%Jz???z
?????) ?Ū??k0? >Z
$

```

署名操作を行った際 PKCS#11API は次に挙げる順番でコールされる。それぞれの PKCS#11 関数の内部でのカードへのアクセス動作について解説する。

1. C\_Initialize : 初期化

初期化処理を行う。この処理の中で、カードインタフェース・モジュールの選択、カード内部のデータを PKCS#15 構造体へマッピングする作業までが完了する。



2. C\_GetSlotList : カードスロットの選択

カードリーダーの一覧を返す関数。カードへのアクセスはない。

3. C\_GetTokenInfo : データの取得

C\_Initialize ですでに読み込み済みの PKCS#15 構造体を含むデータを返す。カードへのアクセスはない。

4. C\_OpenSession : セッションの開始

これもカードへのアクセスはない。

5. C\_Login : ログイン

事前にユーザに入力させた PIN を使用して sc\_pkcs15\_verify\_pin、sc\_pin\_cmd をコールし、最終的に VERIFY カードコマンドを発行する。すでに読み込み済みである暗号情報オブジェクトの中からカード保有者のプライベート鍵を検索、そのプライベート鍵と結びついている AOD を VERIFY に渡している。

6. C\_SignInit : 署名処理の初期化

カードへのアクセスはない。使用するアルゴリズムの決定などを行う。

7. C\_SignUpdate : 署名データの生成

署名対象データをバッファに格納する。対象データのメッセージダイジェスト計算を行う場合はこの関数内で行われる。カードへのアクセスはない。

8. C\_SignFinal : 署名

sc\_pkcs15\_compute\_signature をコールし、その中で sc\_set\_security\_env、sc\_compute\_signature をコールしている。

EF.PrKD から導かれたプライベート鍵実体へのパスを指定して SELECT カードコマンドを発行した後、sc\_set\_security\_env を呼び出すことで、署名に使用するプライベート鍵をカードに対し指示するというのが ISO7816-4 で想定されている動作だが、Cryptoflex の場合、sc\_set\_security\_env 関数内では MANAGE SECURITY ENVIRONMENT カードコマンドも含め、カードコマンドが実行されない。

sc\_compute\_signature 関数内部で PERFORM SECURITY OPERATION カードコマンドの COMPUTE DIGITAL SIGNATURE オペレーションが実行されるかわりに、独自のカードコマンドを実行し、署名作成を行っている。

9. C\_CloseSession : セッションの終了

Cryptoflex カードに依存した処理として、flex\_logout コマンドが実行され、ログ

アウト用のカードコマンドが実行されているが、これは標準で定められた動作ではない。

カードの内容を PKCS#15 構造体にマップする作業は C\_Initialize ですで行われており、その後の操作では認証、署名データ生成する部分でカードへのアクセスが行われる以外は読み込み済みのデータに対する操作が行われるのみである。

つまり、PKCS#11 レイヤーでの署名操作において IC・ID カードが PKCS#15 準拠であることが影響しているのは C\_Initialize のみであることがわかる。

#### 4.1.8. PKCS#15 データモデルの実装とパフォーマンス

PKCS#15 データモデルを実装したカードであっても、専用のデータモデルを採用したカードであっても、データモデルを完全に把握しているミドルウェアからアクセスされる限り、データの選択について処理性能に違いは発生しないはずである。どのファイルに、何の情報が格納されているのかファイルパスと内容の対応をミドルウェアが知っていれば、そのパスを直接指定してアクセスすればよい。

アクセス手順が完了してデータを読み込むとき、PKCS#15 ではデータを ASN.1 表記のデータをエンコードしたフォーマットである、DER フォーマットで格納するため、利用時にはデコードをしなければならない。たとえば、固定長でデータを保存する独自フォーマットを採用した場合と比較して、デコードにかかる処理負荷は大きいといえる。しかし、デコード処理を行うのは IC カードの外側のプログラムであり、その負荷は問題にならないことが多い。

パフォーマンスが低下する、すなわち IC カードに対する処理負荷が高くなるのは、IC カードを利用する外側のプログラム（ミドルウェア）が、IC カード内のデータモデルについて PKCS#15 準拠であるということは知っているが、その内容を完全には把握していない状況でカードの操作を行うときであり、このような場合には IC カードの処理能力が問題になる可能性がある。AID を使って SELECT コマンドを発行、CIA を選択してから MF、EF.CIAInfo、EF.OD を読み込み、DER をデコードして最終的に目的のファイル（多くの場合プライベート鍵への参照や、公開鍵証明書）にたどり着くまでに多くの手順が必要となる。先も書いたとおり DER のデコードは IC カード外部のプログラムが行うため実際には問題とならない場合が多いが、データの読み込み時にカードへの IO が発生するため、この部分でパフォーマンスの低下が起こる可能性がある。

IC カードに PKCS#15 形式でデータを保存すること自体で読み込み処理に時間がか

かるわけではない。データ構造を知らないミドルウェアが、目的とするデータにたどり着くために必要となる情報を、カードから引き出すために時間がかかるのである。

#### 4.1.8.1. 実際のカードで計測(Cryptoflex)

4.1.7 節で使用した Axalto 社の Cryptoflex 32K カードと、Reflex USB v3 カードリーダーを用いて署名操作にかかる実時間を計測する。カードに格納されているデータも 4.1.7 節で使用したものと同一とした。PKCS#11 関数の実行にかかる時間と、カードコマンドの実行に要する時間について gettimeofday(2)関数を利用してマイクロ秒まで計測した。同時に、カードコマンドを APDU として実際にカードに送信する関数である、sc\_transmit\_apdu 関数に費やされる処理時間も計測した。

表 43 PKCS#11 関数の実行と処理時間

PKCS#11 関数	所要時間 ミリ秒：小数第 2 位以下四捨五 入)	発行したカードコマンド数
C_LoadModule	84.0	なし
C_Initialize	1975.5	SELECT * 10 READ BINARY *21
C_GetSlotList	0.1	なし
C_OpenSession	0.0	なし
C_Login	92.6	SELECT * 1 VERIFY * 1
C_SignInit	0.0	なし
C_SignUpdate	0.1	なし
C_SignFinal	338.8	SELECT FILE * 1 署名生成コマンド * 1
C_CloseSession	46.9	(CLOSE * 1)
C_Finalize	0.0	なし
トータル	2674.3	

トータル処理時間のうち、カードと APDU の送受信を行うための sc\_transmit\_apdu

関数にかかった時間は 2369.9 ミリ秒であった。つまり、全処理時間の約 90 パーセントがカードへのアクセスによって費やされていること、カードへのアクセスは C\_Initialize に集中していることがわかる。なお、C\_Initialize 関数内でのカードコマンド実行回数において SELECT よりも READ BINARY の数が多いのは、サイズが大きくて一度に読み込めないデータを複数回の READ BINARY で読み込むためである。

#### 4.1.9. マルチプラットフォーム対応

OpenSC 低レベルライブラリは実際にカードと通信するレイヤーとして OpenCT (Linux, Solaris)、PC/SC (Windows, Mac OS X, Linux)、CT-API (DOS, 組み込み機器など)を利用することができる。プラットフォームの違いはすべてカードリーダー/ライターのレイヤーで吸収されるため、OpenSC 低レベルライブラリ、OpenSC PKCS#15 ライブラリでは違いを意識することはない。しかし、上位レイヤーに位置するアプリケーションインタフェースではプラットフォームごとに異なる実装を提供している。すべてのプラットフォームに共通するインタフェースとして PKCS#11 インタフェースを提供している。Windows 用には PKCS#11 インタフェースのさらに上位に、OpenSC 外で作成されるのだが、Windows Crypto Service Provider が提供されている (Windows CSP)。

Mac OS X 用には tokend と呼ばれる、OpenSC PKCS#15 ライブラリを使用して作成されたプログラムが提供されている。このプログラムは、Mac OS X のセキュリティフレームワークから利用される。tokend は OpenSC プロジェクトで作成したアプリケーションではなく、Mac OS X の基盤をなす OS テクノロジーである Darwin を元に設立されたオープンソースプロジェクト、OpenDarwin.org(<http://www.opendarwin.org>)によるものだが、そのソースコードには OpenSC が使用されている。Mac OS X 上で稼動する Safari browser や Mail client などの Native Mac Applications はセキュリティフレームワークを通じて IC カードとの通信を行う。

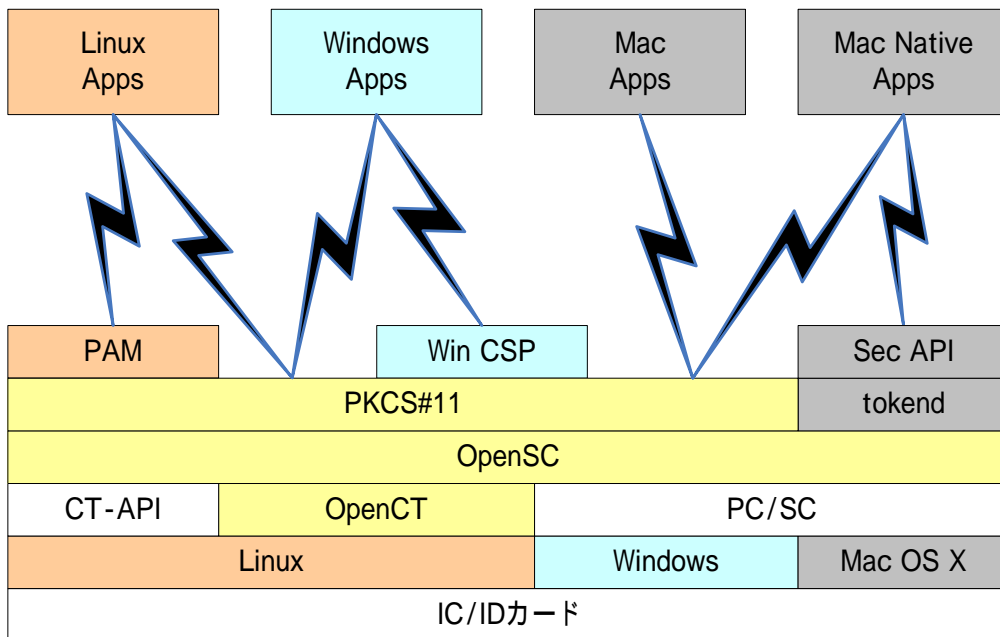


図 38 マルチプラットフォーム対応

## 4.2. PIV

### 4.2.1. 全体のアーキテクチャ

3.4 節で PIV のカードエッジインタフェース、データモデルについて解説した。本節では PIV のミドルウェア参照実装について解説する。ミドルウェア参照実装は、クライアント API を実装する PIV API ライブラリ、カードエッジインタフェースを実装する PIV カードエッジライブラリから構成され、PIV API ライブラリを使用するテストツールが付属している。PIV ではカードコマンドとクライアント API がよく似ているため、ミドルウェアの主な役割はクライアント API とカードコマンドを結ぶことであり、OpenSC のミドルウェアが行っていたようなカードから返されたデータをクライアント API の仕様に会うようにマップする複雑な変換は必要ない。PIV カードエッジライブラリの実装では基本的に引数で渡された値を APDU につめてカードに送信、返された応答を戻す仕事をしている。

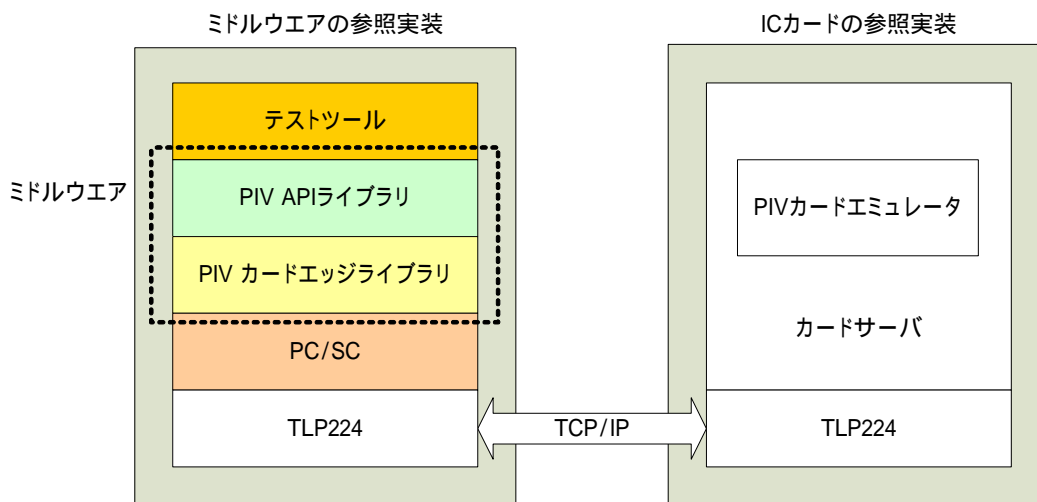


図 39 PIV 参照実装のアーキテクチャ

参照実装のコードに含まれるドキュメントには2005年6月25日の日付がつけられており、バージョンは1.1である。まだ作成途中の部分があり、十分な整備がなされていない印象を受けるが、仕様書をもて理解が難しい部分を把握するためにはとても有用である。

#### 4.2.2. クライアント API 実装

PIV API ライブラリに SP800-73<sup>[21]</sup>に定義されている API がすべて実装されている。

表 44 PIV クライアント API 実装とカードコマンド

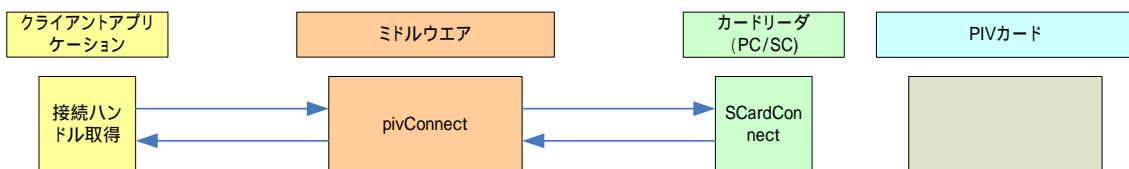
SP800-73 に定義されているクライアント API	参照実装が使用するカードコマンド	意味
pivConnect		接続ハンドル取得
pivDisconnect		接続ハンドル破棄
pivSelectCardApplication	SELECT	カードアプリケーションを選択

pivLogIntoCardApplication	VERIFY GENERAL AUTHENTICATE	カードアプリケーションのセキュリティステータスを確立
pivGetData	GET DATA	
pivLogOutOfCardApplication		カードアプリケーションをリセット
pivCrypt	GENERAL AUTHENTICATE	暗号、署名操作
	CHANGE REFERENCE DATA	パスワード変更
	RESET RETRY COUNTER	リトライカウンタをリセット
pivPutData	PUT DATA	
pivGenerateKeyPair	GENERATE ASYMMETRIC KEY PAIR	非対称鍵ペア生成

カードエッジインタフェースがもつ CHANGE REFERENCE DATA、RESET RETRY COUNTER カードコマンドは使用されておらず、PIN の変更やカウンタのリセット処理を行うための関数はない。

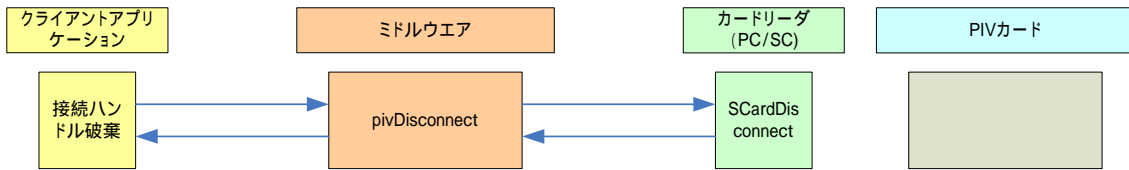
### pivConnect

カードリーダーの初期化、キャッシュデータのクリア等を行っており、カードとの通信は行っていない。



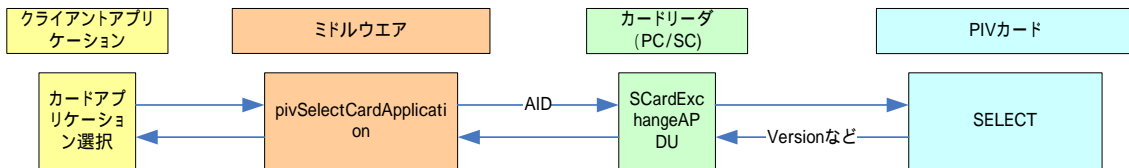
### pivDisconnect

カードリーダーとの接続情報を破棄している。カードとの通信は行っていない。



### pivSelectCardApplication

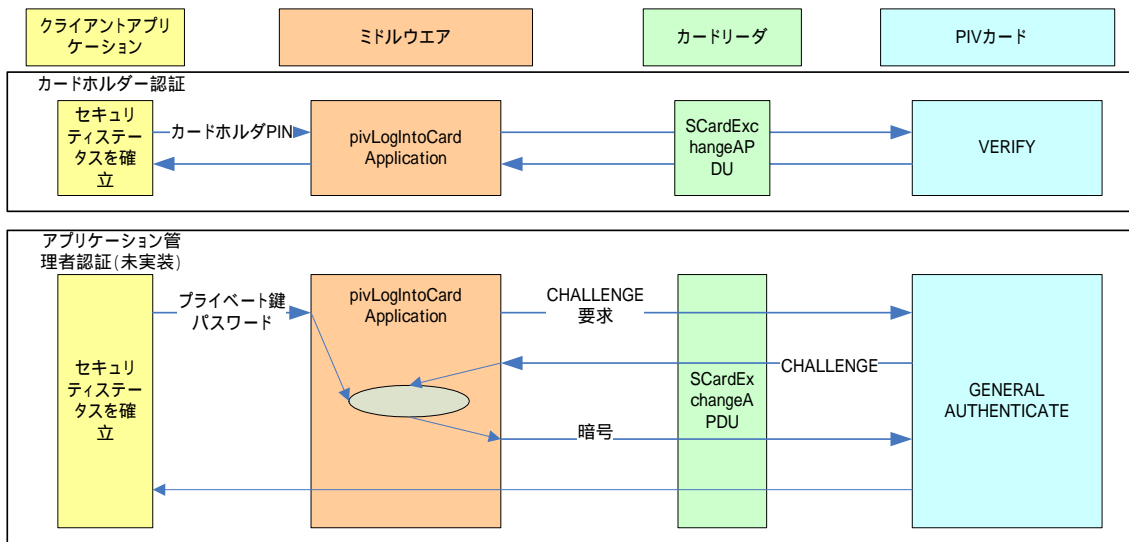
アプリケーション ID を引数に、SELECT カードコマンドを送信している。



### pivLogIntoCardApplication

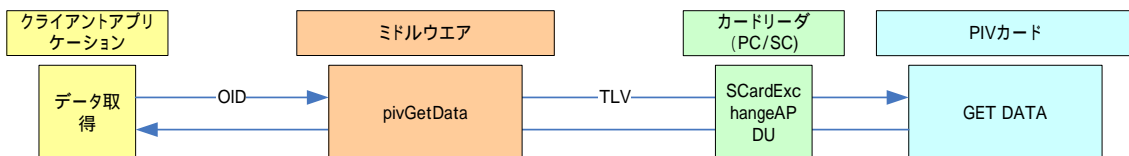
authenticator を引数として受け取る。authenticator は鍵への参照とその PIN を格納したものである。鍵への参照を表す数値は SP800-73 Table12 にて定義されており、Global PIN('00')、Application PIN('80')の2つが、カード保有者を認証するための鍵となっている。pivLogIntoCardApplication 内部で、渡された authenticator の鍵参照が Global PIN、もしくは Application PIN だったとき、VERIFY カードコマンドを実行している。鍵参照がそれ以外るとき、つまり認証される利用者がカード保有者ではなく、カードアプリケーション管理者、またはカードアプリケーション提供者である場合は pivCrypto を実行しており、その中で GENERAL AUTHENTICATE カードコマンドを実行している。pivCrypt の成功、失敗によって pivLogIntoCardApplication の成功、失敗を判断しており、pivCrypt から返された値(暗号データ)は破棄してしまっている。

ここで実行しようとしているのは、おそらく外部認証の一連の手続きなのだが、現在の参照実装では「External Authentication not implemented yet」とあり、正しく実装されていない。



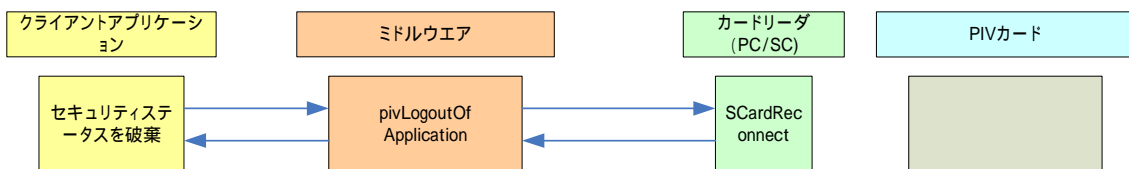
### pivGetData

ハードコードされたテーブルを元にして、引数で与えられた OID から、それに対応するタグデータ（表 26 参照）を取得する。取得したタグデータを引数に GET DATA カードコマンドを実行する。



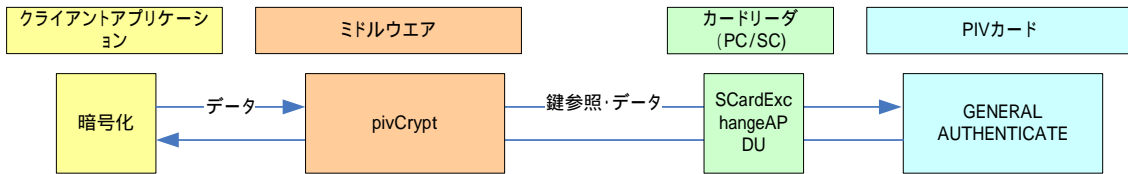
### pivLogoutOfCardApplication

カードリーダーと再接続を行う。これによりカードがリセットされるため、Logout したことになる。



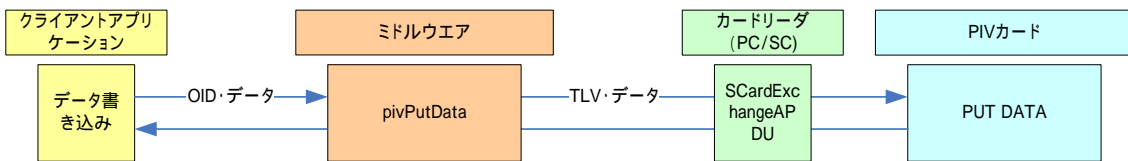
### pivCrypt

内部で GENERAL AUTHENTICATE カードコマンドを実行している。引数はアルゴリズムの種別、鍵への参照、入力データである。



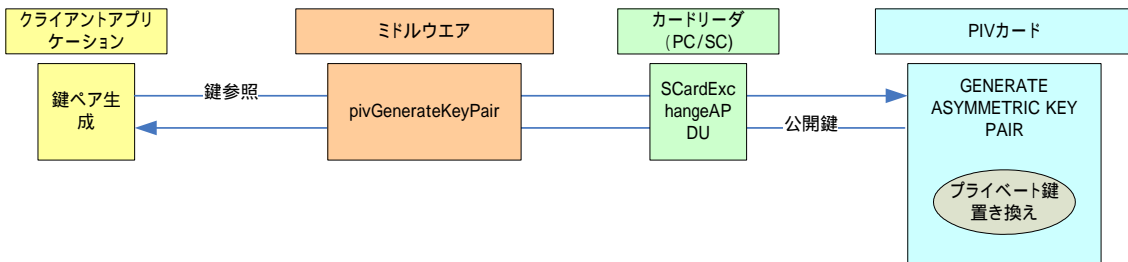
### pivPutData

pivGetData と同様、ハードコードされたテーブルを元に引数で与えられた OID から、それに対応するタグデータ（表 26 参照）を取得する。このタグデータと引数で与えられたデータを引数に、PUT DATA カードコマンドを実行する。



### pivGenerateKeyPair

内部で GENERATE ASYMMETRIC KEY PAIR カードコマンドを実行する。生成された鍵ペアのうち、公開鍵がクライアントアプリケーションまで返される。



全体として、実装が複雑なのは GENERAL AUTHENTICATE を使う認証部分で、他の部分でのミドルウェアの役割は小さい。GENERAL AUTHENTICATE を使う部分でも、カードコマンドの仕様が複雑なのではなく、他と比べて手順や取り決めが複雑なために、実装が難しくなっている。PIV カードは図 13 で示される、汎用カードアクセスレイヤーを IC カードコンポーネントが持つアーキテクチャを採用しており、かつ、汎用カードアクセスレイヤーとミドルウェアが担うサービスアクセスレイヤーとの整合性が高いため、このようにミドルウェアの実装が単純になる。

## 5. テストと認定制度

5章では、IC・IDカードのテストとテストに基づく製品の認定制度の事例を紹介する。具体的には、米国のPIVの事例を示す。PIVについては、3章においてPIVカードとしての説明、4章において、PIVカードに対応したミドルウェアの実装例を説明した。PIVに関連して、3章、4章で説明した文書以外にも大量の技術文書が公開されている。5章では、事例研究としてPIVの背景にあるHSPD-12<sup>[23]</sup>「連邦政府職員と契約業者の共通識別基準のためのポリシー」、FIPS-201「連邦政府職員及び契約業者の個人識別情報の検証」等を説明したのち、PIVのテスト仕様、テストツール、認定制度、認定製品の例等を説明する。

### 5.1. 米国のPIVの成り立ち

#### 5.1.1. HSPD-12

3章、4章の事例研究として説明したPIVは、2004年8月に発令された大統領令HSPD-12「連邦政府職員と契約業者の共通識別基準のためのポリシー」に基づき仕様を作成されている。HSPD-12は、連邦政府施設への物理的・論理的アクセスのセキュリティ強化のため、身分証の標準を規定することを要求しているが、これは、政府機能の効率化のほか、連邦政府職員をテロから守り、また個人情報盗難も防止することを目的としている。この要求に応じたPIVプロジェクトでは、IC・IDカードの発行対象は、連邦政府職員だけでなく、契約業者にも含まれる。そのため非常に多くの枚数が発行されることが見込まれており、2009年までに2000万枚が発行されると言われている。下記にHSPD-12の最初の部分を示す。

主題：連邦政府職員と契約業者の共通識別基準のためのポリシー

(1) テロの標的となりうる連邦政府関係施設へ安全なアクセスを行うための個人識別(Identification)形式に関して、その品質とセキュリティの多様性は排除されるべきである。そのため、合衆国のポリシーとして、安全で信頼できる個人識別形式を政府全体の強制的な標準として確立する。その個人識別形式を連邦政府が職員と契約業者(契約業者の雇用者を含む)に発行することで、セキュリティを高め、政府の効率を上げ、個人(Identity)詐称を減らし、個人のプライバシーを保護する。

(2) パラグラフ(1)で示した政策を実施するために、商務長官は、準拠法に従ってこの指示後6ヶ月を期限として、識別(「基準」)の安全かつ信頼できる形式の連邦政府の基準について、國務長官、国防長官、司法長官、国土安全保障長官、行政管理予算庁(OMB)長官、および科学技術政策局長官と協議した後、発布すること。商務長官は、定期的に基準の見直しを行い、影響を受ける政府機関と協議し、必要に応じて基準をアップデートすること。

HSPD-12の(2)にあるように、指令の実現に向け、商務省(DOC:Department of Commerce)に対し、省庁横断型IDカードの仕組みを構築するための技術標準の作成を、國務省(DOS:Department of State) 国防総省、国土安全保障省や、大統領府行政管理予算局(OMB:Office of Management and Budget)などと協力して行うことを義務付けた。図40にHSPD-12に基づき作成されたるドキュメント類を示す<sup>[50][51]</sup>。

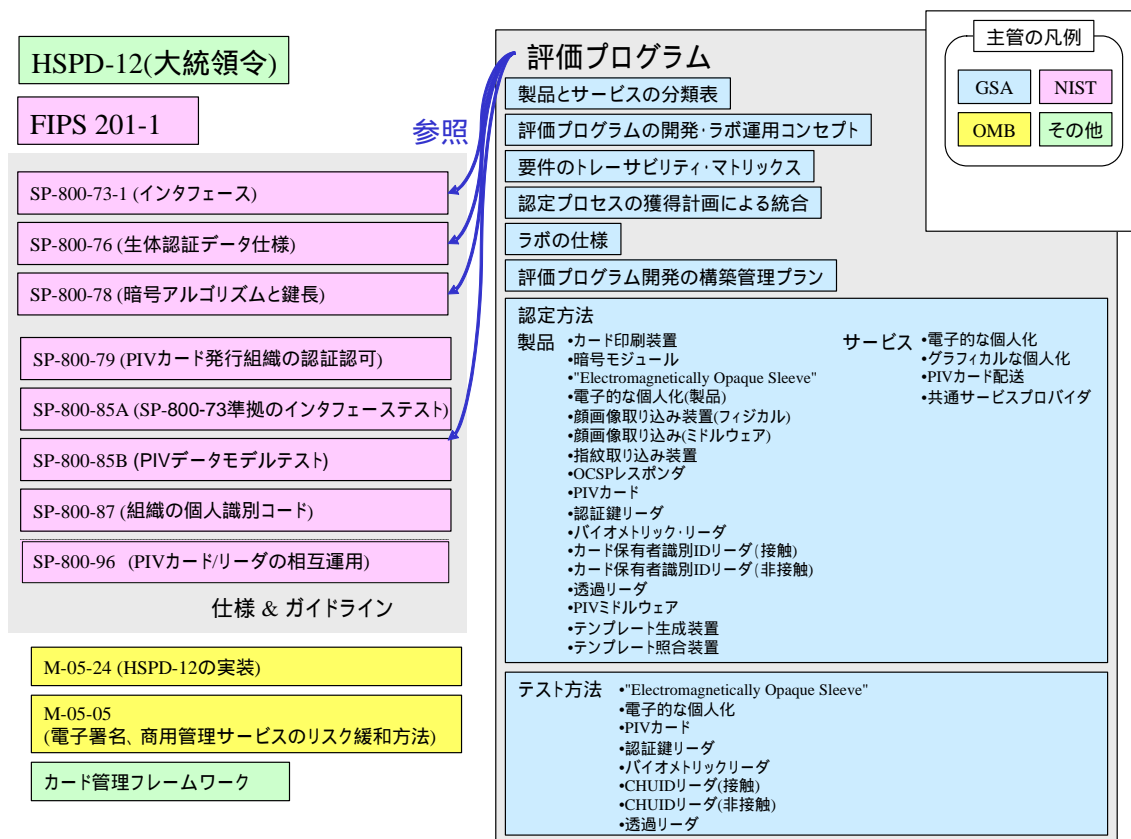


図 40 HSPD-12 に関連した文書

### 5.1.2. FIPS-201

HSPD-12 の指令にある「準拠法に従ってこの指示後 6 ヶ月を期限」に従い商務省長官が HSPD-12 に対応した連邦政府の身分証の基準として発行したのが FIPS-201「連邦政府職員及び契約業者の個人識別情報の検証(Personal Identity Verification (PIV) of Federal Employees and Contractors<sup>[41]</sup>)」である。FIPS 201 は、HSPD-12 が発令された 6 ヶ月後の 2005 年 2 月 25 日に発行された。また、改訂版の FIPS 201-1 が、2006 年 3 月に発行されている。下記に FIPS 201 の序文を示す。

本標準は、連邦政府職員と契約業者の共通の身元確認の標準アーキテクチャと要件の標準を指定する。標準は、アクセス制御に応用されるような個人認証において、信頼でき、費用効率の高いソリューションに要求されるいくつかの問題を扱っている。

全体的なゴールは、連携管理される政府の設備への物理的アクセスと政府の情報システムの電子的アクセスを要求する際に要求された個人の ID を効率のよい検証によって、複数のアプリケーションに対する適切なセキュリティ保証を達成することである。

標準は、解決すべき問題を特定し、共通の ID 検証アーキテクチャを定義し、異なるレベルの保護を要求するアプリケーションに対して必要なセキュリティ保証レベルを達成するのに必要なコンポーネント・インタフェース・支援サービス・ライフサイクル管理機能を記述する。

この標準はまた、ID カード、電気的カードリーダー、通信システム、アクセス制御システムインタフェースの間で相互運用を達成するのに必要である、他の技術的、運用的な標準と結びつき、参照する。

FIPS 201 は、省庁横断型 ID カードの技術標準であり、第 1 部(人物特定、セキュリティ、プライバシーに関する共通の必要条件)と第 2 部(非接触型、バイOMETリクス技術採用といった、詳細な技術仕様)から構成されている。技術標準の作成・調整に当たっては、商務省国立標準技術研究所(NIST)の情報技術研究所(ITL: Information Technology Laboratory)が担当機関となり、技術標準の最終承認は商務省長官が行った。

FIPS 201 は PIV のシステムとしての最小要件と詳細要件を定めている。NIST は、FIPS-201 に基づき、詳細な仕様を作成しているが、これが PIV の仕様書類となっている。図 41 に PIV の仕様に関わる文書の関連図を示す。表 45 に NIST が作成した PIV

に関連した仕様の一覧を示す。

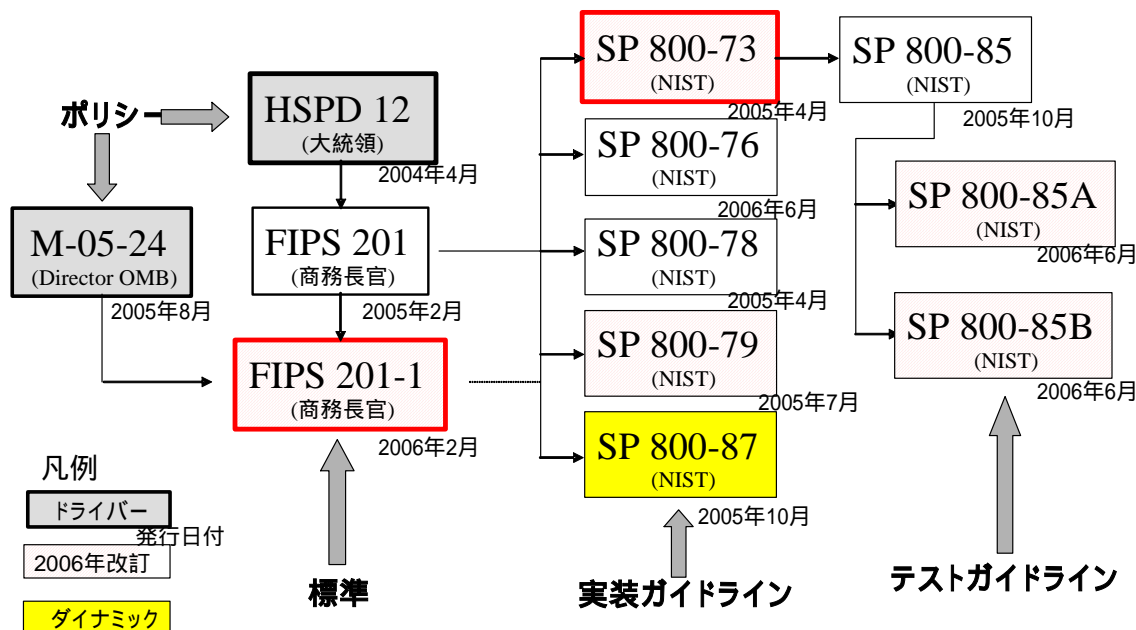


図 41 PIV 技術文書の関連図

表 45 PIV 技術文書の一覧

文書種別と番号	タイトル	概要
FIPS 201-1	連邦政府職員及び契約業者のための個人識別情報の検証 (Personal Identity Verification of Federal Employees and Contractors)	PIV システムにおける最小要件に加え、PIV カードの物理カード特性、ストレージメディア、データ要素について規定された文書
SP 800-73-1	PIV 用のインターフェース (Interfaces for Personal Identity Verification)	証明書の保管・取得に関するインターフェースとカードアーキテクチャ
SP 800-76-1	PIV 用のバイOMETリックデータ仕様 (Biometric Data Specification for Personal Identity Verification)	バイOMETリック情報のインターフェースとデータフォーマット

SP 800-78-1	PIV 用の暗号アルゴリズムと鍵長 (Cryptographic Algorithms and Key Sizes for Personal Identity Verification)	PIV で利用される暗号アルゴリズムや鍵長、パラメータなどの仕様
SP 800-79 <sup>[45]</sup>	PIV カード発行組織の認証と認定のガイドライン (Guidelines for the Certification and Accreditation of PIV Card Issuing Organizations)	認定を望むカード発行者が実施すべきいくつかの属性についての記述
SP 800-85A <sup>[46]</sup>	PIV カードアプリケーションとミドルウェアインターフェースのテストガイドライン (PIV Card Application and Middleware Interface Test Guidelines (SP 800-73 compliance))	PIV ミドルウェアと PIV カードアプリケーションの 2 つへの準拠検証に使われるべきテスト要件とテストアサーションについて記述
SP 800-85B <sup>[47]</sup>	PIV データモデルテストガイドライン (PIV Data Model Test Guidelines)	データモデルに関するテスト要件や、テストアサーション、準拠テストについてのガイドライン
SP 800-87 <sup>[48]</sup>	連邦政府組織・関連組織識別用コード (Codes for the Identification of Federal and Federally Assisted Organizations)	FIPS 201 で規定されているカード保有者 ID (CHUID) に含まれる連邦政府カード証明書番号 (FASC-N) を記述するのに必要な組織コードの一覧
SP 800-96	PIV Card / Reader Interoperability Guidelines	PIV とカードリーダ間の追う相互運用可能性を確保するためのガイドライン

NISTIR 7284	Personal Identity Verification Card Management Report,	SP 800-78-1 には PIV カード管理の仕様は含まれていない。カード管理システムの概要を示すと同時に、今後の PIV の仕様変更、追加を示唆している。
NISTIR 7337 <sup>[52]</sup>	Personal Identity Verification Demonstration Summary	NIST が企画した PIV カード等、FIPS-201 をサポートした製品のデモンストレーションの報告書。HSPD-12/FIPS-201 の実装状況を報告している

### 5.1.3. HSPD-12 実施のロードマップ

前述したとおり、FIPS 201 は、HSPD-12 が発令された 6 ヶ月後の 2005 年 2 月 25 日に発行された。FIPS 201 と FIPS 201 に基づいた IC・ID カード（サービス）の仕様である SP 800-73 の原案は、2004 年 11 月 8 日に発表されている。注目すべきは、この原案に対して 80 を超える個人と組織から 1900 以上のパブリックコメントが寄せられたことである (<http://csrc.nist.gov/piv-program/FIPS201-Public-Comments.html>)。こうしたパブリックコメントを反映したのち FIPS 201、SP800-73 が正式に発行されている。

連邦政府の身分証の基準である FIPS-201 の発行後、行政管理予算局（OMB）がより詳細な連邦 ID カード導入に向けた計画指針を M-05-24 として発表した。M-05-24 では、HSPD-12 の実現に向けたロードマップが示され、各連邦機関は、FIPS-201 の第 1 部（人物特定、セキュリティ、プライバシーに関する共通の必要条件）の採用を 2005 年 10 月 27 日に、第 2 部（非接触型、バイOMETRICS 技術採用といった、詳細な技術仕様）の採用を 2006 年 10 月 27 日までに完了することが求められた。表 46 に M-05-24 で示されたロードマップと主な仕様の発行日を示す。

表 46 HSPD-12 のマイルストーンと主な文書の発行日

日付	マイルストーン 発行ドキュメント	内容 (タイトル)
2004 年 8 月 27 日	HSPD-12	大統領令 HSPD-12 「連邦政府職員と契約業者の共通識別基準のためのポリシー」
2004 年 11 月 8 日		FIPS-201 と SP800-73 の最初の原案が発表される
2005 年 2 月 25 日	FIPS-201	「連邦政府職員及び契約業者の個人識別情報の検証」の発行
2005 年 4 月 8 日	SP800-73	PIV 用のインターフェース
2005 年 4 月 25 日	SP800-78	PIV 用の暗号アルゴリズムと鍵長
2005 年 8 月 5 日	M-05-24	大統領令 HSPD-12 の実装のガイドライン。 HSPD-12 の実現に向けたロードマップ
2005 年 10 月 27 日		FIPS-201 技術標準第 1 部順守期限
2006 年 2 月 1 日	SP800-76	PIV 用のバイOMETリックデータ仕様
2006 年 4 月 5 日	SP 800-85A	PIV カードアプリケーションとミドルウェアインターフェースのテストガイドライン
2006 年 7 月 27 日	SP 800-85B	PIV データモデルテストガイドライン
2006 年 10 月 27 日		FPS-201 技術標準第 2 部順守期限 PIV カードの発行開始。

## 5.2. PIV のテスト方法論と仕様

HSPD-12 と M-05-24 により、各米国連邦機関は、FIPS-201 の基準を満足した PIV カードの発行を期限付きで求められた。重要な基準のひとつが PIV カードの相互運用可能性の確保ということになる。NITS では、FIPS-201 に基づき、様々な詳細な仕様を作成してしているが、その中に PIV のテスト仕様がある。単にテスト仕様を提供するだけでなく、テストを行うためのレファレンス実装や、テストツール、更に FIPS-201 の基準を満足した PIV カードとミドルウェアであることを認定するための制度等を提供している。

### 5.2.1. テスト方法論

PIV カードとミドルウェアの仕様は、SP800-73 に記述されている。この詳細は、3章、4章で紹介している。この SP800-73 の仕様に対応したテスト仕様書として作成されたのが SP800-85 である。SP800-85 は、テストの目的に従い SP800-85A、SP800-85B に分冊して作成されている。

PIV カードとミドルウェアは、後述するうに認定制度があり、各連邦政府機関は、認定された PIV カードとミドルウェア製品を購入することになる。SP800-85A は、この製品を認定するためのテスト仕様書として利用される。

各連邦政府機関は、認定された PIV カードとミドルウェアを購入して、このカードを各連邦機関の個々の要求に応じたオプションの選択した上でカードのパーソナライズを行なう。この各連邦政府機関のパーソナライズに対応したテストのための仕様書が SP800-85B ということになる。図 42、表 47 に SP800-73 と SP800-85、更に各政府機関の関係を示す。NPIVP については 5.3 節で説明する。

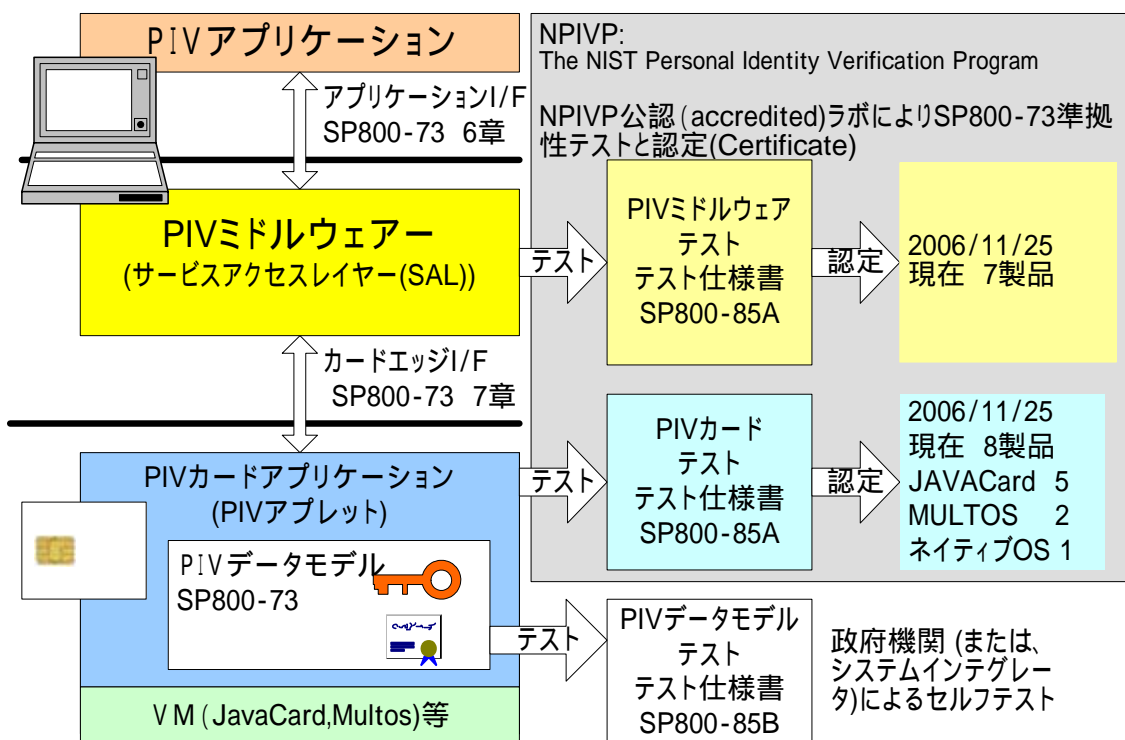


図 42 SP800-73 と SP800-85 の関係

表 47 PIV カード、ミドルウェアとテスト仕様の関係

テスト対象	誰によるテスト	何時テストされるか	テスト仕様書
PIV ミドルウェア I/F	公 認 さ れ た (Accredited) NPIVP ラボ	政府機関の調達前	SP 800-85A
PIV カードアプリケーション I/F	公 認 さ れ た (Accredited) NPIVP ラボ	政府機関の調達前(カ ードのパーソナライ ゼーション以前)	SP 800-85A
PIV データモデル (カード上のコンテン ツ)	政府機関 (または、シ ステムインテグレー タ)	カード発行の間 (パーソナライゼーシ ョン)	SP 800-85B

### 5.2.2. SP800-85A

SP800-85A は PIV カードアプリケーション I/F、つまりカードエッジインタフェースをテストするための仕様と、PIV ミドルウェア I/F、アプリケーションインタフェースをテストするための仕様を策定したドキュメントである。

ドキュメントのターゲットは PIV カードアプリケーション開発者、PIV ミドルウェア開発者である。PIV カードサービスを利用するために必要となる、バックエンドシステムへのアクセス制御、カード発行システム、カードリーダー、生体認証用ドライバなどのテストについては範囲外であるとしている。

テスト環境は PC で稼動するテストツールキット、接触および非接触カードリーダー(またはデュアルインタフェースカードリーダー)、PIN パッドなどの PIN 入力装置、PIV カードとミドルウェアによって構成され、ミドルウェアのテストを行う際には SP800-73 に準拠した PIV カード、もしくは PIV カードエミュレータを使用する必要がある。

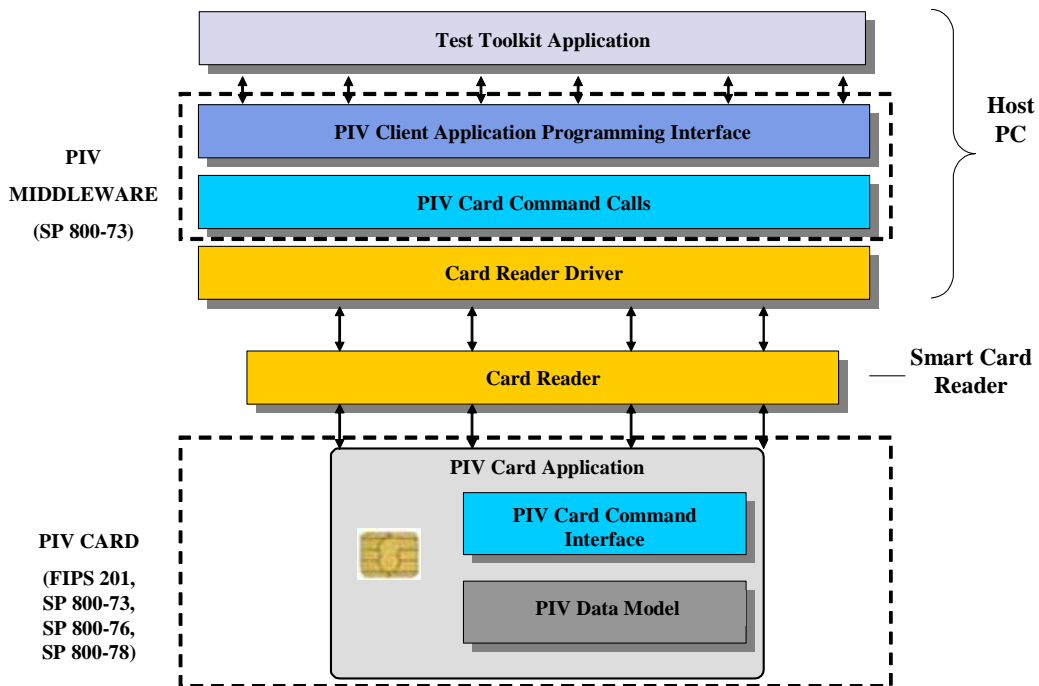


図 43 SP800-85A テストアーキテクチャ

ミドルウェアのテストではアプリケーションインタフェースの関数ごとにテストケースを用意しており、それぞれのテストケースでテストの目的、対象関数、テスト手順などを明示している。

表 48 SP800-85A API テスト

テスト対象関数	テストケース数	テスト内容
pivConnect	6	接続に関するテスト。2度続けて接続したときは接続に失敗する動作など。
pivCrypt	7	データ暗号化に関するテスト。カードへのログインが終了していない、指定する鍵参照やアルゴリズムが不正な場合など。
pivDisconnect	4	接続終了に関するテスト。未接続なコネクションを切断しようとしたときに失敗する動作など。

pivGenerateKeyPair	5	鍵生成に関するテスト。生成対象鍵のリファレンス番号や、暗号方法などを変えて行うテストなど。
pivGetData	6	データの取得に関するテスト。ログインが不要、必要なデータの取得や、存在しない OID 指定によるテストなど。
pivLogIntoCardApplication	4	選択済みカードアプリケーションへのログイン(セキュリティステータスの確立)に関するテスト。間違った PIN 入力時の動作など。
pivLogOutOfCardApplication	3	ログアウトのテスト。ログアウト後もコネクションは有効であることのテストなど。
pivPutData	4	データ書き込みに関するテスト。アプリケーション管理者権限でのログインが必要である点についてのテストなど。
pivSelectCardApplication	4	カードアプリケーションの選択に関するテスト。AID をすべて指定したとき、省略形式で指定したときのテストなど。

カードエッジインタフェースのテストはカードコマンドひとつにつき、接触、非接触の場合それぞれひとつずつ用意されている。APDU が正しく送受信できていることが前提であり、テストケースには「GET DATA コマンドを送信する」「69 82'が返ってくる」というレベルの説明がなされている。

### 5.2.3. SP800-85B

SP800-85B は PIV カードに格納されるデータモデルが FIPS201、SP800-73、SP800-76、SP800-78 にマッチしていることをテストするためのテスト仕様である。データの取得、格納手順などは SP800-85A でテストされ、SP800-85B では純粋にデータ自体に関するテストのみを扱う。

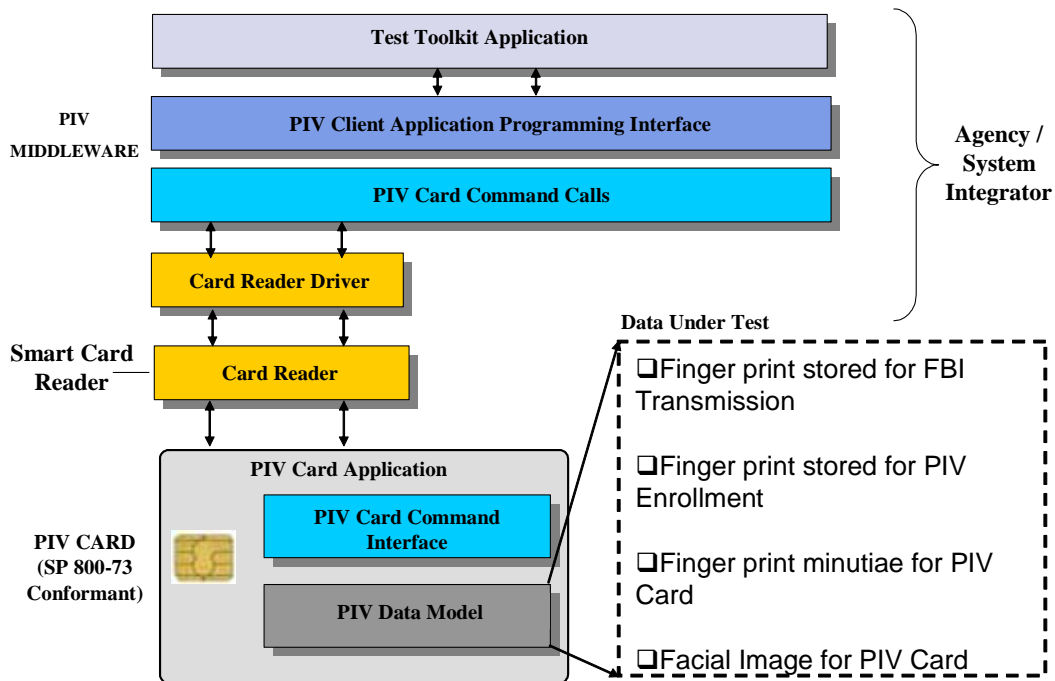


図 44 SP800-85B テストアーキテクチャ

TLV ( Tag-Length-Value ) 形式で扱われるデータ全般、生体認証データ、署名済みデータ、PKI 証明書についてそれぞれデータが従うべきルールと、その根拠となる標準が説明されている。データひとつずつに従うべきルールが記述されているのではなく、はじめに4つに分類されたルールを説明し、次に対象データとルールのマップを提供して、データが従うべきルールを示すことで説明が簡潔になっている。ルールは全体で179定義されている。

表 49 SP800-85B のテストルール

ルール種別	ルール数	概要
BER-TLV に関するルール	9	データのエンコーディング自体のほか日付のフォーマットやデータサイズに関する制限事項など。
生体認証データに関するルール	50	生体認証情報のヘッダ情報、指紋、顔認証用のデータについての取り決めなど。

署名済みデータに関するルール	75	データフォーマットが Cryptographic Message Syntax external digital signature(RFC3852)である、SignedData であるなど。
非対称鍵ペアに関するルール	45	ハッシュアルゴリズムが SHA256 であること、署名用の鍵の keyUsage が digitalSignature および nonRepudiation のみであることなど。

#### 5.2.4. PIV のテストツール

現在テストに利用可能なツールとして、NIST が提供する参照実装、テストデータ、テストデータ生成ツールがある。参照実装<sup>[53]</sup>は Microsoft VisualStudio のプロジェクトとして提供されており、PIV カードをエミュレートするカードエミュレータプロジェクトと、ミドルウェアとコマンドラインツールからなるミドルウェアプロジェクトから構成される。ミドルウェア部分は通常 TLP224 (TCP/IP を使ってカードリーダーと通信するためのプロトコル) ライブラリを利用してカードエミュレータと通信するよう構成されているが、TLP224 マクロを定義せずにコンパイルすれば PC/SC ライブラリをコールすることもできる。参照実装は PIV テスト用に準備されたプログラムではなく、ミドルウェアのアプリケーションインタフェースやカードエッジインタフェースをテストするためのツールは特に用意されていない。参照実装に「pivTest」というコマンドが含まれているが、これは参照実装を稼働させるためのプログラムであり、PIV ミドルウェアをテストするためのものではない。すべてのソースが公開されているため、先に紹介した SP800-85A,B を実行するための変更を行うのは容易である。

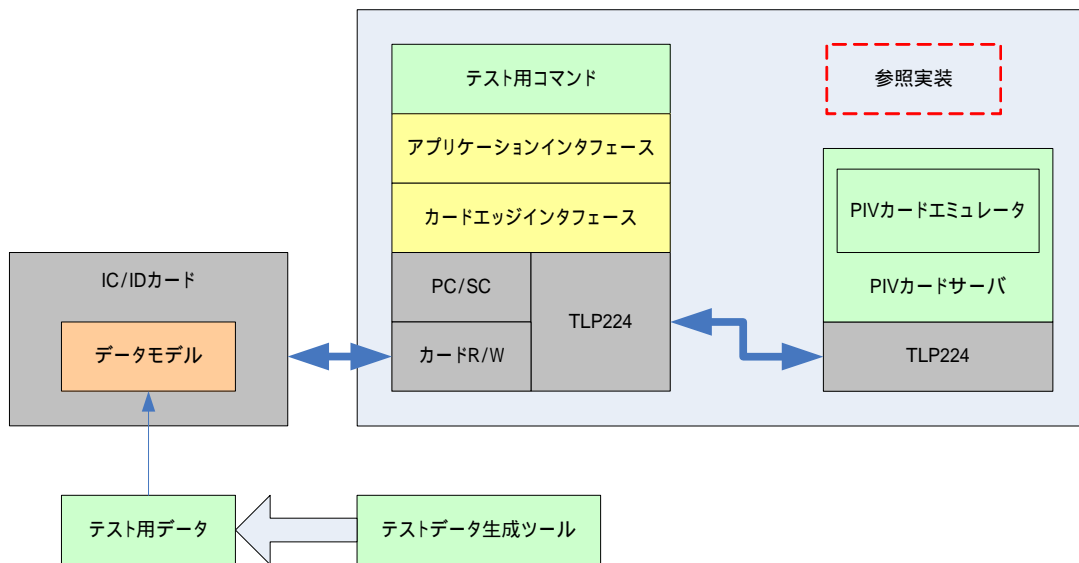


図 45 PIV テストツール

NIST が提供するテスト用データは、これも NIST が提供するテストデータ生成ツールによって作られたもので、CCC (Card Capability Container) 指紋認証データ、顔認証データのほか、カード所有者の証明書、アプリケーション管理者、鍵管理者の証明書やセキュリティオブジェクトなどが別々のファイルに DER フォーマットで格納されている。

テストデータ生成ツールは Java で作られた GUI アプリケーションで、任意の CHUID、CCC、証明書データや、生体認証データを作成することができる。ただし、フォーマットが不正であるとか、keyUsage に nonRepudiation を含む認証用証明書といった、エラーとなるべきデータを生成する機能は持っていないため、テストケースの実行に必要なデータのすべてを生成することはできない。

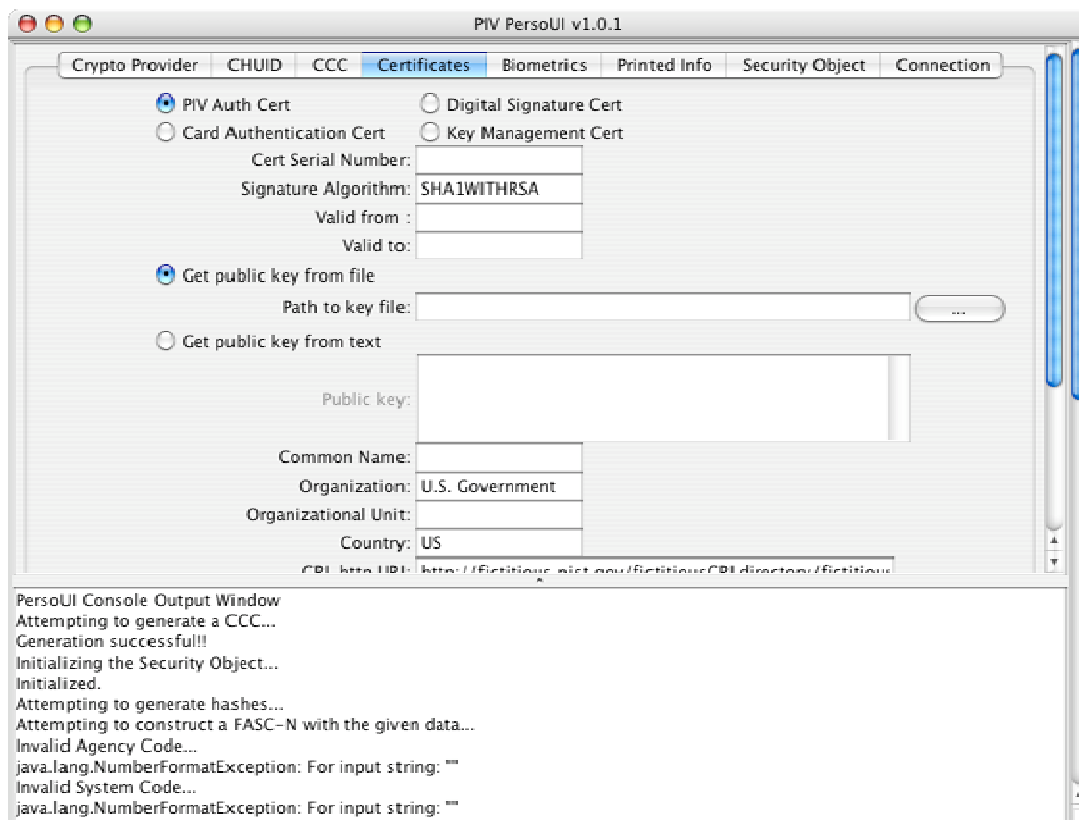


図 46 データ生成ツールの画面

### 5.3. PIV の認定制度と製品

#### 5.3.1. PIV の認定制度

HSPD-12 の実装、及び、FIPS-201 の基準に関連した製品、サービスは多岐に渡っているが、これらの製品やサービスの評価(Evaluation)と認定(Approval)を行なう必要があった。NIST と調達局(GSA: General Services Administration)は、HSPD-12 の実装のために必要となる製品やサービスのテストと評価のためのプログラムを確立した。

NIST は、FIPS-201 に準拠した PIV カードとミドルウェアを検証して、認定する NPIVP(NIST Personal Identity Verification Program)を立ち上げた。

NPIVP は、PIV カード (PIV カードエッジインターフェースを実装した PIV カードアプリケーション) とミドルウェアが SP800-73 の仕様を満足しているかのテストと検証を提供している。テスト仕様としては、SP 800-85A が使われテストが行われる。図 47 に PIV ミドルウェアのテスト、検証、認定の手順を示す。

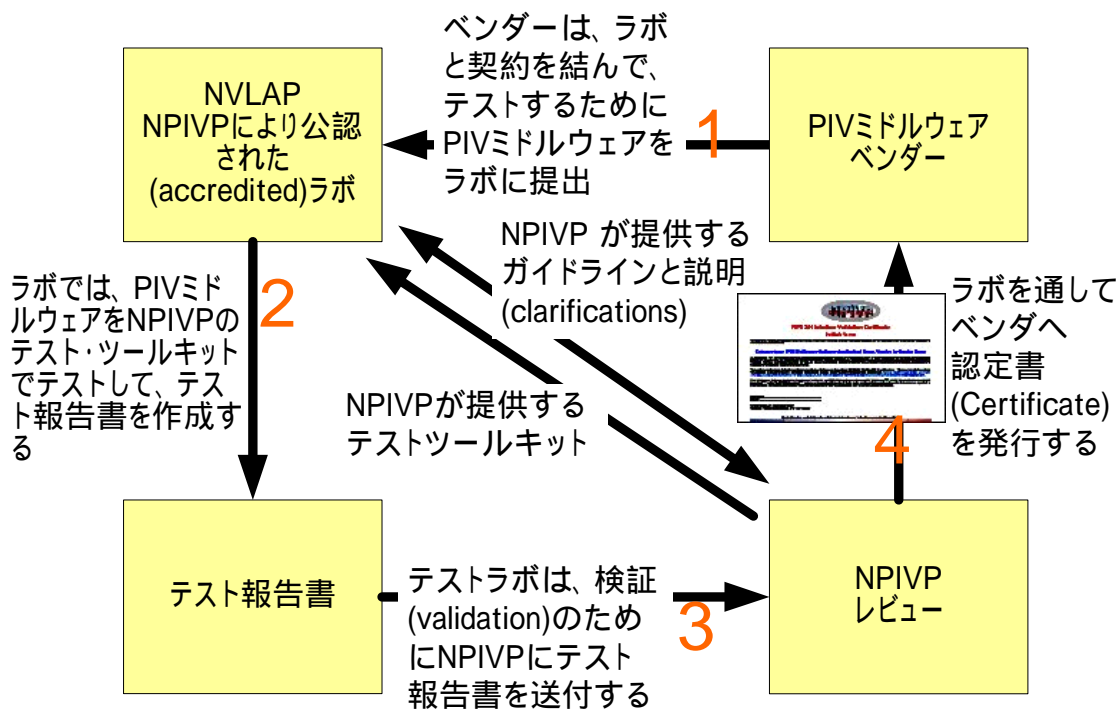


図 47 PIV ミドルウェアの検証と認定

図 47 にあるようにテスト自体は、NPIVP により公認された (Accredited) ラボにより行なわれる。

PIV カードに関しては、NPIVP による認定だけでなく、FIPS-140 の認定取得が必要になる。FIPS-140 の認定は、FIPS-140 に基づいた暗号モジュール評価制度である CMVP (Cryptographic Module Validation Program) により行なわれる。FIPS-140 は、米国政府機関が暗号製品を購入する場合の調達基準あり、CMVP は、FIPS-140 適合製品を認定する制度ということになる。CMVP は、これまでも数多くの暗号製品の認定を行なってきており十分な実績を積んでいる。また、CMVP に似た制度は、日本においても JCMVP として IPA において試行が開始されている。NPIVP も CMVP を参考にした制度だと考えられ非常に似た制度となっている。

FIPS-140 は、米国政府機関が暗号製品を調達する際の基準であるが、実際には、産業界においても暗号製品のデファクトの基準となっている。暗号製品の評価の難しさ故、客観的な評価がもとめられ、CMVP のような評価・認定制度が重要な意味を持つようになった。米国連邦政府機関の調達がドライブ役となり導入が推進されその結果、産業界においても暗号製品のデファクトの基準となった経緯がある。同様に、IC・ID カードの相互運用可能性を確保した製品も同じように非常に評価が難しい。複雑な標準の展開は、

標準仕様からだけのアプローチでは成し得ないかもしれず、NPVIP のような評価・認定制度がデファクトの標準を作り出す可能性がある。

NPVIP において行なわれるテストと製品検証の状況と結果は、NIST NPVIP ウェブサイト(<http://csrc.nist.gov/npivp/>)において公表される。

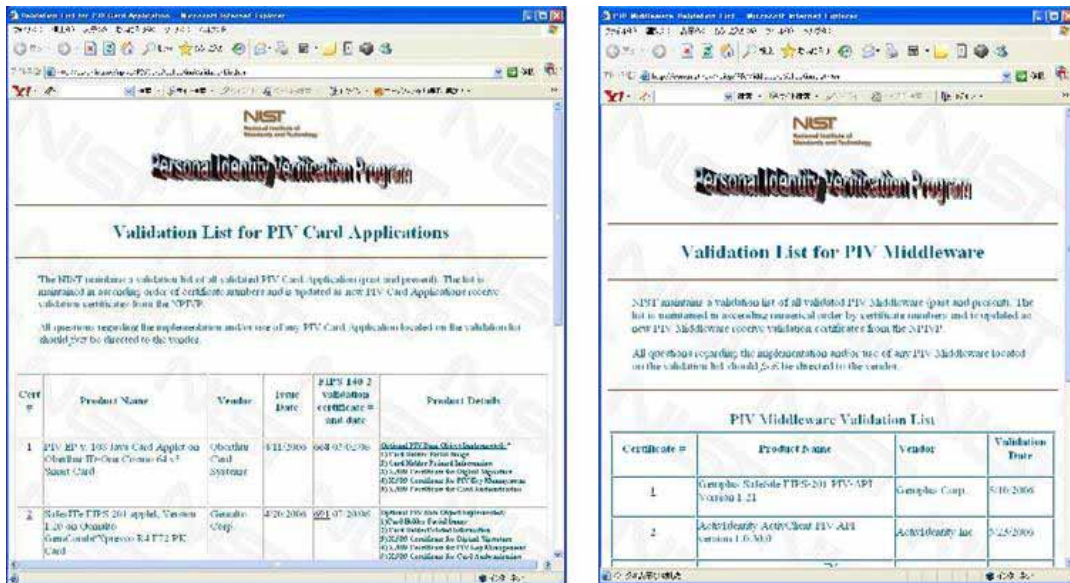


図 48 NPVIP ウェブサイトにおける認定状況の公表

PIV カードについては、NPVIP と CMVP の認定を取得したのち、PIV ミドルウェアについては NPVIP の認定を取得したのち FIPS-201 認定製品となる。FIPS-201 認定製品は、GSA の FIPS-201 評価制度ウェブサイト (<http://fips201ep.cio.gov/>, <http://fips201ep.cio.gov/apl.php>)において、他の認定製品と共に公表される。

### 5.3.2. PIV の認定製品

2006 年 11 月現在、NPVIP ウェブサイトにおいて PIV カードアプリケーションリストについて 8 製品、PIV ミドルウェア製品に関しては、7 製品が記載されている。PIV カードアプリケーションリストにある 8 製品の中で、FIPS-140 も取得済みの製品は 3 製品となっている。

PIV カードアプリケーションリストについて 8 製品において、カード OS としては、javaCard をベースとした製品が 5 製品、Multos カードをベースとした製品が 2 製品、ネイティブなカード OS で PIV カードエッジインターフェースを実現した製品は 1 製品

となっている。表 50 に 2006 年 11 月現在での PIV カードの認定製品を示す。

表 50 PIV カードアプリケーションリスト

認定番号 認定日	製品名	ベンダー 名	備考
No.1 2006 年 4 月 11 日	PIV EP v. 108 Java Card Applet on Oberthur ID-One Cosmo 64 v5 Smart Card	Oberthur Card Systems	FIPS-140 No.668(2006/5/2) Java カード <a href="http://csrc.nist.gov/cryptval/140-1/140sp/140sp668.pdf">http://csrc.nist.gov/cryptval/140-1/140sp/140sp668.pdf</a>
No.2 2006 年 4 月 20 日	SafesITe FIPS 201 applet, Version 1.20 on Gemalto GemCombiXpresso R4 E72 PK Card	Gemalto Corp.	FIPS-140 No.691(2006/7/20) Java カード <a href="http://csrc.nist.gov/cryptval/140-1/140sp/140sp691.pdf">http://csrc.nist.gov/cryptval/140-1/140sp/140sp691.pdf</a>
No.3 2006 年 5 月 26 日	PIV Application on Hitachi MULTOS Smart Card, Hardware Version: AE45X1, Firmware Version 1.0	Hitachi, Ltd.	Multos
No.4 2006 年 6 月 6 日	SETECS Inc.'s OneCARDTM PIV-II Java Card Applet (Version 1.2) on Gemalto GemCombi Xpresso R4 E72 PK card	SETECS Inc	FIPS-140 No.705(2006/9/20) Java カード <a href="http://csrc.nist.gov/cryptval/140-1/140sp/140sp705.pdf">http://csrc.nist.gov/cryptval/140-1/140sp/140sp705.pdf</a>
No.5 2006 年 7 月 31 日	StepNexus PIV Application v4.2.1 on Keycorp MULTOS 64K Smart Card	Keycorp Limited	Multos
No.6 2006 年 9 月 5 日	PIV card application version 19 on SafeNet Smart Card 400 (72K) SCCOS version 3.0	SafeNet Inc.	ネイティブ OS <a href="http://www.safenet-inc.com/Library/3/SmartCard_400.pdf">http://www.safenet-inc.com/Library/3/SmartCard_400.pdf</a>

No.7 2006年 10月20日	ActivIdentity PIV End-Point Applet version 2.6.2 on Oberthur ID-One Cosmo 64 v5	ActivIdentity Inc.	Java カード
No.8 2006年 11月11日	PIV Applet version 01 on J-IDMark 64 PIV	Sagem Orga Inc.	Java カード

## 参考文献リスト

- [1] ISO/IEC JTC 1, “ISO/IEC 7816-1:1998 Identification cards -- Integrated circuit(s) cards with contacts -- Part 1: Physical characteristics”, Nov. 2003
- [2] ISO/IEC JTC 1, “ISO/IEC 7816-2:1999 Identification cards -- Integrated circuit cards -- Part 2: Cards with contacts -- Dimensions and location of the contacts”, 1999
- [3] ISO/IEC JTC 1 “ISO/IEC 7816-3:2006 Identification cards -- Integrated circuit cards -- Part 3: Cards with contacts -- Electrical interface and transmission protocols”, Oct. 2006
- [4] ISO/IEC JTC 1 “ISO/IEC 14443-1:2000 Identification cards -- Contactless integrated circuit(s) cards -- Proximity cards -- Part 1: Physical characteristics”, 2000
- [5] ISO/IEC JTC 1 “ISO/IEC 14443-2:2001 Identification cards -- Contactless integrated circuit(s) cards -- Proximity cards -- Part 2: Radio frequency power and signal interface”, 2001
- [6] ISO/IEC JTC 1 “ISO/IEC 14443-3:2001 Identification cards -- Contactless integrated circuit(s) cards -- Proximity cards -- Part 3: Initialization and anticollision”, 2001
- [7] ISO/IEC JTC 1 “ISO/IEC 14443-4:2001 Identification cards -- Contactless integrated circuit(s) cards -- Proximity cards -- Part 4: Transmission protocol”, 2001
- [8] ISO/IEC JTC 1, “ISO/IEC 7816-4:2005 Identification cards -- Integrated circuit cards -- Part 4: Organization, security and commands for interchange”, Jan. 2005
- [9] RSA Laboratories, “PKCS #11: Cryptographic Token Interface Standard”, Jun. 2004, <ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf>
- [10] 独立行政法人情報処理推進機構 セキュリティセンター, “セキュリティ API に関する技術調査”, 2004年8月, [http://www.ipa.go.jp/security/fy15/reports/sec\\_api/](http://www.ipa.go.jp/security/fy15/reports/sec_api/)
- [11] T Schwarzhoff, J. Dray, et al, “Government Smart Card Interoperability Specification Version 2.1”, NIST Interagency Report 6887, Jul. 2003, <http://csrc.nist.gov/publications/nistir/nistir-6887.pdf>
- [12] 情報処理振興事業協会セキュリティセンター, “本人認証技術の現状に関する調査”, 2003年7月, <http://www.ipa.go.jp/security/fy14/reports/authentication/index.html>

- [13] RSA Laboratories, "PKCS #15 v1.1: Cryptographic Token Information Syntax Standard", Jun. 2000, [ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1\\_1.pdf](ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-15/pkcs-15v1_1.pdf)
- [14] ISO/IEC JTC 1, "ISO/IEC 7816-15:2004 Identification cards -- Integrated circuit cards -- Part 15: Cryptographic information application", Jan. 2004
- [15] 日本工業標準調査会 審議, "IC カード - 第 15 部 : 暗号情報アプリケーション JIS X 6320-15:2006 (ISO/IEC 7816-15:2004)", 2006 年 2 月
- [16] ISO/IEC JTC 1, "ISO/IEC FDIS 24727-1 Identification cards -- Integrated circuit card programming interfaces -- Part 1: Architecture", Nov. 2005
- [17] ISO/IEC JTC 1, "ISO/IEC FCD 24727-2 Identification cards -- Integrated circuit card programming interfaces -- Part 2: Generic card interface", Dec. 2005
- [18] ISO/IEC JTC 1, "ISO/IEC CD 24727-3 Identification cards -- Integrated circuit card programming interfaces -- Part 3: Application interface", Dec. 2005
- [19] ISO/IEC JTC 1, "ISO/IEC NP 24727-4 Identification Cards -- Programming Interfaces for Integrated Circuit Cards -- Part 4: API administration", Dec. 2005
- [20] ISO/IEC JTC 1, "ISO/IEC NP 24727-5 Identification Cards -- Programming Interfaces for Integrated Circuit Cards -- Part 5: Testing", Dec. 2005
- [21] J. Dray, et al., "Interfaces for Personal Identity Verification" NIST Special Publication 800-73-1, Apr. 2006,  
<http://csrc.nist.gov/publications/nistpubs/800-73-1/sp800-73-1v7-April20-2006.pdf>
- [22] F. Maes, "European Electronic Identity Practices: Country Update of Belgium", Porvoo 10 Conference on Interoperable European Electronic Identities, Nov. 2006,  
[http://porvoo10.net/p10/14\\_Country\\_update\\_Porvoo10\\_Belgium.ppt](http://porvoo10.net/p10/14_Country_update_Porvoo10_Belgium.ppt)
- [23] G. W. Bush, "Homeland Security Presidential Directive/Hspd-12", Aug. 2004,  
<http://www.whitehouse.gov/news/releases/2004/08/20040827-8.html>
- [24] T. Polk, et al., "Cryptographic Algorithms and Key Sizes for Personal Identity Verification", NIST Special Publication 800-78, Apr. 2005,  
<http://csrc.nist.gov/publications/nistpubs/800-78/sp800-78-final.pdf>
- [25] ISO/IEC JTC 1, "ISO/IEC 7816-10:1999 Identification cards -- Integrated circuit(s) cards with contacts -- Part 10: Electronic signals and answer to reset for synchronous cards", 1999
- [26] ISO/IEC JTC 1, "ISO/IEC 7816-12:2005 Identification cards - Integrated circuit cards -- Part 12: Cards with contacts -- USB electrical interface and operating procedures", 2005

- [27] ISO/IEC JTC 1, "ISO/IEC 7816-5:2004 Identification cards -- Integrated circuit cards -- Part 5: Registration of application providers", 2004
- [28] ISO/IEC JTC 1, "ISO/IEC 7816-7:1999 Identification cards -- Integrated circuit(s) cards with contacts -- Part 7: Interindustry commands for Structured Card Query Language (SCQL)", 1999
- [29] ISO/IEC JTC 1, "ISO/IEC 7816-8:2004 Identification cards -- Integrated circuit cards -- Part 8: Commands for security operations", Jun. 2004
- [30] ISO/IEC JTC 1, "ISO/IEC 7816-9:2004 Identification cards -- Integrated circuit cards -- Part 9: Commands for card management", Jun. 2004
- [31] ISO/IEC JTC 1, "ISO/IEC 7816-11:2004 Identification cards -- Integrated circuit cards -- Part 11: Personal verification through biometric methods", 2004
- [32] Population Register Centre (VRK), "FINEID S1 - Electronic ID Application, v2.1", Mar. 2003,  
[http://www.fineid.fi/vrk/fineid/files.nsf/files/4A6480742C01D98BC2257054002A1D23/\\$file/S1v21.pdf](http://www.fineid.fi/vrk/fineid/files.nsf/files/4A6480742C01D98BC2257054002A1D23/$file/S1v21.pdf)
- [33] Population Register Centre (VRK), "FINEID S4-1 – FINEID Implementation profile 1 for Finnish Electronic ID Card, v2.1A", Oct. 2004,  
[http://www.fineid.fi/vrk/fineid/files.nsf/files/E99479C66998368DC2257054002A8686/\\$file/S4-1v21A.pdf](http://www.fineid.fi/vrk/fineid/files.nsf/files/E99479C66998368DC2257054002A8686/$file/S4-1v21A.pdf)
- [34] Population Register Centre (VRK), "FINEID S2 - VRK (PRC) CA-model and certificate contents, v2.1", Jul. 2005,  
[http://www.fineid.fi/vrk/fineid/files.nsf/files/24EA4C4CD4A1EAA0C2257054002A55BD/\\$file/S2v21.pdf](http://www.fineid.fi/vrk/fineid/files.nsf/files/24EA4C4CD4A1EAA0C2257054002A55BD/$file/S2v21.pdf)
- [35] Population Register Centre (VRK), "FINEID S4-2 - FINEID Implementation profile 2 for Organizational Usage, v2.1A", Oct. 2004,  
[http://www.fineid.fi/vrk/fineid/files.nsf/files/85365C8B0A87FAA4C2257054002C2A29/\\$file/S4-2v21A.pdf](http://www.fineid.fi/vrk/fineid/files.nsf/files/85365C8B0A87FAA4C2257054002C2A29/$file/S4-2v21A.pdf)
- [36] Population Register Centre (VRK), "FINEID S5 - Directory Specification, v2.1", Oct. 2004,  
[http://www.fineid.fi/vrk/fineid/files.nsf/files/070D5F2D1228DF78C22570540034FDA5/\\$file/S5v21.pdf](http://www.fineid.fi/vrk/fineid/files.nsf/files/070D5F2D1228DF78C22570540034FDA5/$file/S5v21.pdf)
- [37] CEN, "CWA 14890-1 Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic requirements", Mar. 2004,

- <ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/eSign/cwa14890-01-2004-Mar.pdf>
- [38] CEN, "CWA 14890-2 Application Interface for smart cards used as Secure Signature Creation Devices - Part 2: Additional Services", May 2004,  
<ftp://ftp.cenorm.be/PUBLIC/CWAs/e-Europe/eSign/cwa14890-02-2004-May.pdf>
- [39] D. D. Cock, et al., "The Belgian Electronic Identity Card (Overview)", In Sicherheit 2005: Sicherheit - Schutz und Zuverlässigkeit, Beiträge der 3rd Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.v. (GI), Lecture Notes in Informatics (LNI) LNI P-77, J. Dittmann (ed.), Bonner Köllen Verlag, pp. 298--301, 2006, <http://www.cosic.esat.kuleuven.be/publications/article-769.pdf>
- [40] "Belpic Chip Specifications", Nov. 2002,  
[http://www.rijksregister.fgov.be/cie\\_fr/cie/archives\\_diverses/cahiers\\_charge/eik\\_best\\_ek\\_bijlage5.doc](http://www.rijksregister.fgov.be/cie_fr/cie/archives_diverses/cahiers_charge/eik_best_ek_bijlage5.doc)
- [41] NIST, "FIPS PUB 201-1 Personal Identity Verification (PIV) of Federal Employees and Contractors", Jun. 2006,  
<http://csrc.nist.gov/publications/fips/fips201-1/FIPS-201-1-chng1.pdf>
- [42] C. Wilson, et al., "Biometric Data Specification for Personal Identity Verification", NIST Special Publication 800-76, Feb. 2006,  
<http://csrc.nist.gov/publications/nistpubs/800-76/sp800-76.pdf>
- [43] J. Dray, D. Corcoran, "Personal Identity Verification Card Management Report", NIST Interagency Report 7284, Jan. 2006,  
<http://csrc.nist.gov/publications/nistir/ir7284/nistir-7284.pdf>
- [44] OpenSC project, <http://www.opensc-project.org/>
- [45] D. Branstad, et al., "Guidelines for the Certification and Accreditation of PIV Card Issuing Organizations", NIST Special Publication 800-79, Jul. 2005,  
<http://csrc.nist.gov/publications/nistpubs/800-79/sp800-79.pdf>
- [46] R. Chandramouli, et al., "PIV Card Application and Middleware Interface Test Guidelines (SP800-73 compliance)", NIST Special Publication 800-85A, Apr. 2006,  
<http://csrc.nist.gov/publications/nistpubs/800-85A/SP800-85A.pdf>
- [47] R. Chandramouli, et al., "PIV Data Model Test Guidelines", NIST Special Publication 800-85B, Jul. 2006,  
<http://csrc.nist.gov/publications/nistpubs/800-85B/SP800-85b-072406-final.pdf>
- [48] W. C. Barker, et al., "Codes for the Identification of Federal and Federally Assisted Organizations", NIST Special Publication 800-87, Jan. 2006,

- <http://csrc.nist.gov/publications/nistpubs/800-87/sp800-87-Final.pdf>
- [49] J. Dray, et al., "PIV Card to Reader Interoperability Guidelines", NIST Special Publication 800-96, Sep. 2006,  
<http://csrc.nist.gov/publications/nistpubs/800-96/SP800-96-091106.pdf>
- [50] Office of Management and Budget, "M-05-24, Implementation of Homeland Security Presidential Directive (HSPD) 12 - Policy for a Common Identification Standard for Federal Employees and Contractors", Aug. 2005,  
<http://www.whitehouse.gov/omb/memoranda/fy2005/m05-24.pdf>
- [51] Office of Management and Budget, "M-05-05, Electronic Signatures: How to Mitigate the Risk of Commercial Managed Services", Dec. 2004,  
<http://www.whitehouse.gov/omb/memoranda/fy2005/m05-05.pdf>
- [52] E. McCallister, et al., "Personal Identity Verification Demonstration Summary", NIST Interagency Report 7337, Aug. 2006,  
[http://csrc.nist.gov/publications/nistir/NISTIR-7337\\_CRADA\\_082006.pdf](http://csrc.nist.gov/publications/nistir/NISTIR-7337_CRADA_082006.pdf)
- [53] Archive PIV Reference Implementation and Data Generator Source Code,  
<http://csrc.nist.gov/piv-program/archive-piv-ridgsc-.html>
- [54] Open Source for eID Interoperability Open Source for eID Interoperability Status and Demonstrator  
[http://www.fineid.fi/vrk/fineid/files.nsf/files/DE091D21E5644CFAC22570A700450084/\\$file/09\\_porvoo8-bruegger18\\_Italy.pdf](http://www.fineid.fi/vrk/fineid/files.nsf/files/DE091D21E5644CFAC22570A700450084/$file/09_porvoo8-bruegger18_Italy.pdf)
- [55] Open Source Solutions for Interoperability  
[http://porvoo7.fjarmalaraduneyti.is/media/Porvoo7/Open\\_source\\_solutions\\_for\\_eID.doc](http://porvoo7.fjarmalaraduneyti.is/media/Porvoo7/Open_source_solutions_for_eID.doc)
- [56] PIV のカードエッジインタフェース仕様を満足するフリーの実装コード  
<http://www.identityalliance.com/downloads/PIV-II-MSU.zip>  
<http://lists.musclecard.com/pipermail/muscle/2005-September/004387.html>
- [57] 情報処理振興事業協会, "「情報セキュリティ分野における技術ロードマップ～ICカードシステムにおける情報セキュリティ～」報告書", 2006年3月  
<http://www.ipa.go.jp/about/pubcomme/200603/060315RoadmapHokoku.pdf>
- [58] 松本泰, "社会システムとしての電子認証と電子署名", 2005年12月  
[http://www.jnsa.org/jnsapress/vol15/15\\_03-11.pdf](http://www.jnsa.org/jnsapress/vol15/15_03-11.pdf)
- [59] 独立行政法人情報処理推進機構 セキュリティセンター, "SP800-73-1 翻訳", 2005年4月, <http://www.ipa.go.jp/security/publications/nist/documents/SP800-73-J.pdf>

[60] 独立行政法人情報処理推進機構 セキュリティセンター, “SP800-76 翻訳”, 2006 年 2 月, <http://www.ipa.go.jp/security/publications/nist/documents/SP800-76-J.pdf>