JAHIS標準 08-002

HPKI 対応 IC カードガイドライン

2008 年6月

保健医療福祉情報システム工業会

カードシステム委員会

HPKI 対応 IC カードガイドライン

ま　え　が　き

本ガイドラインは保健医療福祉分野における電子署名を行うに際して利用される PKI の機能を搭載した IC カード及び IC カードの利用環境に対する要求事項を定めたものである。また本ガイドラインは、別途制定が進められている「JAHIS HPKI 電子署名規格 V1.0」とともに利用されることを前提としており、同規格の下で利用される IC カードの相互運用性の仕様も定めている。

平成 12 年に「電子署名及び認証業務に関する法律」が成立し、日本において電子的な署名が認められて以来、電子署名は電子契約などの分野において徐々に活用されつつある。保健医療福祉分野においても、平成 17 年 3 月に厚生労働省により「医療情報システムの安全管理に関するガイドライン」が策定され、署名・押印が義務付けされた文書等を電子的に作成する際において電子署名を代替に用いる場合及び e-文書法に対応して、スキャナ等により電子保存する場合について電子署名の基準が明記された。また、同年 4 月には、同省にて「保健医療福祉分野 PKI 認証局 証明書ポリシ」が策定され、国際標準に準拠した保健医療福祉分野向けの PKI（HPKI）の発行ルールが確定した。また、IT 新改革戦略においても HPKI の推進が明記され、普及に向けた各種施策が行われているところである。

JAHIS は、産業界の業界団体として、これら国の施策に協力するとともに、普及促進を図るための相互運用性の確保を図ることが重要な役割であることから、今般、「JAHIS HPKI 対応 IC カードガイドライン」を策定することとし、ここに JAHIS 標準として公開するものである。

本ガイドラインは、JAHIS 会員各社の意見を集約し、「JAHIS 標準」の一つとして発行したものである。したがって、会員各社がシステムの開発・更新に当たって、本規格に基づいた開発・改良を行い、本規格に準拠していることをその製品のカタログ・仕様書等に示し、さらにその製品の使用においてユーザが理解すべき内容を説明する場合などに使われることを期待している。

なお、本規格で扱う電子署名およびシステムの要件は、参照規格や技術動向にあわせて変化する可能性がある。JAHIS としても継続的に本規格のメンテナンスを重ねてゆく所存であるが、本規格の利用者はこのことにも留意されたい。

本ガイドラインが、HPKI の普及・推進に多少とも貢献できれば幸いである。

2008 年 6 月

保健医療福祉情報システム工業会
カードシステム委員会

＜＜ 告知事項 ＞＞

本ガイドラインは関連団体の所属の有無に関わらず、ガイドラインの引用を明示することで自由に使用することができるものとします。ただし一部の改変を伴う場合は個々の責任において行い本ガイドラインの準拠する旨を表現することは厳禁するものとします。

本ガイドラインならびに本ガイドラインに基づいたシステムの導入・運用についてあらゆる障害や損害について、本ガイドライン作成者は何らの責任を負わないものとします。ただし、関連団体所属の正規の資格者は本ガイドラインについての疑義を作成者に申し入れることができ、作成者はこれに誠意をもって協議するものとします。

目次

第１章　適応範囲

本ガイドラインでは、電子署名を目的とした HPKI で使用される IC カード、及び IC カードの利用環境に対する要求事項を定める。

－ IC カード機能・仕様
－ IC カードのセキュリティ要件
－ IC カードを利用する端末の機能
－ IC カードを利用する端末のセキュリティ要件
－ 相互運用性を確保するための IC カード内の PKI アプリケーションの仕様
－ 相互運用性を確保するための IC カードを利用する際のインタフェースの仕様

第２章　引用規格・引用文献

厚生労働省・保健医療福祉分野 PKI 認証局　証明書ポリシ（以降 HPKI 証明書ポリシと記述する）　http://www.mhlw.go.jp/shingi/2005/04/s0401-1.html

JIS X 19790:2007 セキュリティ技術 – 暗号モジュール　有効のセキュリティ要求事項

ISO/IEC 19790:2006 Information technology – Security techniques – Sequrity requirements for Cryptographic modules.

JIS X 6320-4:200X IC カード—第 4 部: 交換のための構成､セキュリティ及びコマンド（出版予定）

JIS X 6320-8:2006 IC カード—第 8 部: セキュリティ処理コマンド

JIS X 6320-15:2006 IC カード—第 15 部　暗号情報アプリケーション

ISO/IEC 7816-4:2005 Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange

ISO/IEC 7816-8:2004 Identification cards – Integrated circuit cards – Part 8: Commands for security operations

ISO/IEC 7816-15:2004 Identification cards – Integrated circuit cards – Part 15: Cryptographic information application

Crypto API　http://msdn2.microsoft.com/en-us/library/aa380252.aspx

PKCS #11 V2.20　http://www.rsa.com/rsalabs/node.asp?id=2124

第３章　用語の定義

DF (dedicated file)

ファイル制御情報と任意選択として割付可能なメモリとを含んでいる構造体。

EF (elementary file)

同一ファイル識別子によって識別されるデータオブジェクト、レコード又はデータ単位の集合体

HPKI (Helathcare Public Key Infrastructure)

保健医療分野のために構築された公開鍵認証基盤で、医療従事者の資格等の属性を証明する機能を有する

IC カード

１つ以上の IC チップを搭載したカード。形状は ISO/IEC 7810 によって規定される

IEF　(internal elementary file)

カード内にテータを格納する EF であって、暗号鍵のように、カードが内容を解釈する。

PIN (Personal Identification Number)

一般には特定の機能を使用する際に認証を得るためにエンティティを確認するために入力される数字の文字列を指すが、本ガイドラインではパスワードと同じく数字以外の文字を含んだ文字列を指す

PKI (Public Key Infrastructure)

エンティティの信頼性に関して、信頼できる第三者が公開鍵に対して電子署名を付与した公開鍵証明書によって保障をする非対称暗号技術を応用した認証基盤

WEF（working elementary file）

カード内にデータを格納する EF であって、カードが内容を解釈しない

オフセット

データ単位を使用する EF のデータ単位を連続して参照する数値。

公開鍵

エンティティの非対称鍵対のうち、公にされて使用される鍵

私有鍵

エンティティの非対称鍵対のうち、そのエンティティにだけによって使用されるべき鍵

証明書

人等のエンティティとその関連する公開鍵を関連付けるディジタル署名によって保護された文書

認証局

公開鍵認証基盤で、エンティティに対して公開鍵証明書を発行することによってエンティティの存在を保証する組織体

ハッシュ値

あるデータが与えられた場合に、ハッシュ関数（暗号的に安全性の確立された一方向性関数）によって得られたそのデータを代表する数値

ファイル識別子（file identifier）

ファイルを指定するために指定される２バイトのデータ要素。

第４章　HPKI 用 IC カードの機能

４．１　IC カードの種類

４．１．１　ネイティブ型 IC カード

ネイティブ OS を搭載する IC カードをネイティブ型 IC カードと呼ぶ。ネイティブ OS とは、IC カード内のすべてのソフトウェアが IC チップに依存するプログラムコードで構成される OS をいう。また、ネイティブ OS は、外部ノードから IC カードに送られるすべてのコマンドを、OS レベルで解釈実行することを特徴とする。また、そのため実行処理速度が速いことが特徴である。通常、固定メモリのネイティブ型 IC カードはカードが発行された後は、アプリケーション機能の追加、変更および削除が困難であるが、メモリをセクタ管理するものには可能なものが存在する。しかし、このタイプは次に述べるプラットフォーム型 IC カードの範疇に含まれるものとする。

４．１．２　プラットフォーム型 IC カード

プラットフォームを搭載する IC カードを、プラットフォーム型 IC カードと呼ぶ。プラットフォームとは、外部ノードから IC カードに送られるコマンドを解釈・実行するアプリケーションソフトウェア（カードアプリケーション）をダウンロードすることが可能な IC カードに搭載される OS を含むソフトウェアをいう。また、プラットフォームは、IC カード内にダウンロードされたカードアプリケーションを、SELECT FILE コマンドによって選択、起動し、以降、外部ノードから IC カードに送られるコマンドを、他のカードアプリケーションを選択する場合を除き、当該カードアプリケーションで処理することを特徴とする。代表的なプラットフォームとして、JavaCard、MULTOS がある。JavaCard、MULTOS のようにカードアプリケーションが IC カード内の IC チップに依存しないプログラムコードによって構成される場合もある。このような場合、プラットフォームは、IC チップに依存しないプログラムコードを、IC チップに依存するプログラムコードに変換する仮想マシン（VM）を有する。さらに広義には、IC カード内のすべてのソフトウェアが IC チップに依存するプログラムコードで構成される場合であっても、カードアプリケーションをダウンロード可能であり、SELECT FILE コマンドでカードアプリケーションを選択、起動し、以降外部ノードからのコマンドをカードアプリケーションが処理することを特徴と

する機能を実現するソフトウェアの場合、プラットフォームと呼ぶ。
プラットフォーム型 IC カードは カードが発行された後でも、アプリケーション機能がプログラムのダウンロードによって追加、変更が可能であり、不必要になれば削除が可能である。

４．２　IC カードアプリケーションの構成

４．２．１　シングルアプリケーション

一つの IC カードのＣＰＵ上で１種類のサービスしか行わないカードをシングルアプリケーションカードという。初期の交通系の IC カード等がこれに該当する。現在の金融系の IC カードは、IC キャッシュカード、デビッドカード、静脈認証など複数のアプリケーションが搭載されているので、この範疇には属さない。

４．２．２　マルチアプリケーション

マルチアプリケーションカードは 1 枚のカード上で複数のサービスを提供することが可能なカードをいう。さらに、IC カードのアプリケーションを発行後に追加する管理モデルが、次世代 IC カードシステム研究会（NICSS）より提案されており、NICSS モデルと呼ばれる。　NICSS モデルは、カード発行者(Card Issuer)とサービス提供者(Service Provider)が独立であること、サービス提供者が独自のアプリケーションを追加可能である点が特徴となる。住民基本台帳カードや行政連携カードなど、複数のカード発行者とサービス提供者が存在する公共サービスなどで導入が検討されている。

４．３　HPKI 用 IC カードに要求される機能

４．３．１　私有鍵保存機能

私有鍵（Private Key）は公開鍵と対になる鍵である。私有鍵は、その加入者によって秘密保持すべき情報であり、公開せず、他人に漏れないように鍵の所有者だけが管理する。認証局は、いかなる場合でもこれらの鍵へのアクセス手段を提供しない。（HPKI 証明書ポリシ）
署名のために使用される私有鍵は、法律によって必要とされる場合を除き、預託されないものとする。また、署名目的の私有鍵の回復も行わない。（HPKI 証明書ポリシ）
加入者の私有鍵が認証局で生成される場合は、IETF RFC 4210「証明書管理プロトコル」に従ってオンライントランザクションで、又は同様に安全な方法によって、加入者に引き渡されるものとする。認証局はオリジナルの私有鍵を引き渡した後は私有鍵のコピーを所有していないことの証明ができるものとする。（HPKI 証明書ポリシ）
IC カード内で非対称の鍵ペアを生成する場合は GENERATE ASYMMETRIC KEY PAIR コマンドにより，カード内で生成し、内部基礎ファイル（IEF）に格納する。（JIS X 6320-8：2006）
私有鍵はハッシュ値を入力とした署名演算に使用される。

４．３．２　公開鍵証明書保存機能

加入者の公開鍵が加入者により生成される場合は、IETF RFC 4210「証明書管理プロトコル」に従ってオンライントランザクションで、又は同様に安全な方法によって、認証局に引き渡され、公開鍵証明書が加入者に送付され、IC カード内の WEF に格納される。公開鍵の検証時には IC カードから読み出される。

４．３．３　私有鍵生成機能（オプション）（含む公開鍵エクスポート機能）

GENERATE ASYMMETRIC KEY PAIR コマンドにより，　IC カード内で鍵ペアを生成し、私有鍵は内部基礎ファイル（IEF）に格納される。また、公開鍵証明書をＣＡ局へ要求するために GENERATE ASYMMETRIC KEY PAIR コマンドにより、既に IC カード内で生成された公開鍵にアクセスする。(JIS X 6320-8：2006)

４．３．４　私有鍵インポート機能（オプション）

鍵の IC カード内での生成は望ましいが必須ではない。外部で鍵ペアを生成した場合は私有鍵を IEF へ、公開鍵証明書を WEF インポートする。

４．１．５　公開鍵証明書エクスポート機能

IC カード内に保存された公開鍵証明書は公開鍵の検証のために署名に付属させる場合等に WEF を指定して読み出す。

４．３．６　署名機能

PERFORM SECURITY OPERATION コマンドの COMPUTE DIGITAL SIGNATURE 処理により，ディジタル署名の計算を行う。
署名計算の中で使用する私有鍵を指定し，ハッシュ値が IC カードに入力される。

４．３．７　私有鍵活性化機能

私有鍵を署名などの運用に使用することができる状態にすることを活性化（Activate）という。私有鍵の活性化データが認証局で生成される場合は、活性化データが加入者に伝えられた後は、認証局においては完全に破棄し、保管しないものとする。また、伝えられた活性化データは、認証局で定められた規定に従い、加入者により安全に保護するものとする。
私有鍵の活性化データを加入者が生成する場合は、認証局で定められた規定に従い、加入者により安全に保護するものとする。（HPKI 証明書ポリシより）
私有鍵の活性化の手段としては、加入者による利用者認証、生体認証、端末が使用する IC カードがドメインで認められたものかを IC カードが認証する端末認証などがあるが、本ガイドラインでは、PIN 照合による利用者認証のみを規定している。
加入者確認の本人確認の為の認証鍵と署名鍵を同じものを用いると、署名時のハッシュ値を認証時にチャレンジコードとして IC カードに送り込むとレスポンスコードとして署名された値が自動的に送付されてしまうので、認証用の鍵と署名鍵は分けなくてはならない。

署名は本来、署名する内容に対して確認し意識して署名をするもので、そのために署名に対しては必ず、加入者の意思確認のための確認として私有鍵の活性化ステップを入れなくてはならない。

４．４　IC カードのセキュリティ機能

４．４．１　アプリケーションの安全性

１）偽造・複製が困難

耐タンパ性によってカードに内蔵された IC チップ上のの情報を保護している。具体的には、以下のような複製を行うための情報取得が困難な技術的な対策によって保護している。

- 樹脂を封入してカードからチップを取り出しするのが難しくする。
- IC チップを多層化することによって顕微鏡での観察困難にする
- 異常に利用されたことを検出するセンサーを組み込み、異常を検知するとデータを消去するなど電気的な解析を困難にする。
- 情報を記録するメモリをランダムに配置することで、再現を困難にする。
- 消費電力や電圧の変化からのデータ解析を防ぐため、消費電力や計算時間等を均一あるいはランダムにする

２）不正使用が困難

暗証番号・パスワードあるいは暗号鍵の確認によって、正当性を IC カードが確認したアクセスだけが許可される仕組みを持っている。アクセス制御は、アプリケーション単位、ファイル単位に設定することが可能で、読出し・書込み・書換えなどの操作ごとに異なるアクセス権を設定することが可能となっている。

暗証番号・パスワードおよび暗号鍵は、値を設定できるが読み出すことのできない情報として IC カード内で管理されるので、カード製造者や管理者であっても不正な利用によってその値を取得することはできない。暗証番号・パスワードあるいは暗号鍵の確認は、あらかじめ設定された回数の照合や認証に失敗すると鍵の利用を自動的に停止（閉塞）することが可能なので、不正な利用を多数回繰り返すことも難しい仕組みを持っている。

３）アプリケーションの独立性

複数のアプリケーションが１枚の IC カード上に存在する場合、各々のアプリケーション内のファイルや鍵、セキュリティ属性は独立である。そのため、あるアプリケーション内のデータへの操作が別のアプリケーションのデータに影響を与えることはない。また、あるアプリケーション内の認証結果によるセキュリティ状態が、別のアプリケーションに引き継がれることはない。そのためあるアプリケーションへの操作が別のアプリケーションに影響を与えることはなく、それぞれのアプリケーションがあたかも独立な IC カードであるように安全性を保ったまま動作させることが可能である。

４．４．２　カードアプリケーションに要求されるセキュリティ機能

１）端末認証（外部認証機能）

利用される端末の正当性を外部認証の機能によって確認する機能。 端末が鍵を知って

いることによる端末の正当性を確認する。例えば、GET CHALLENGE コマンドと、それに続く EXTERNAL AUTHENTICATE コマンドを使用する。サーバアクセスするサーバの認証にも利用できる。（注）

２） 利用者認証

１）PIN の照合により PIN を知っていることによる利用者の確認。例えば VERIFY コマンドを使用する。

２） 鍵を知っている（持っている）ことによる利用者の確認。例えば、GET CHALLENGE コマンドと、それに続く EXTERNAL AUTHENTICATE コマンドを使用する。

３） セキュアメッセージング

コマンド及びその応答に対してその一部又は全体を暗号によって保護するための方法(ISO/IEC 7816-4)。コマンド実行のとき通信されるデータを暗号によって保護する。データの隠蔽機能と完全性チェック機能を持つことが出来る。端末と IC カード間での通信の安全性が保証されない場合に用いられる通信の安全性確保の機能である。（注）

４） 内部認証

カード（アプリケーション）の正当性を端末が認証するための機能である。接続端末装置から送られるチャレンジを、INTERNAL AUTHENTICATE コマンドにより IC カード内に格納されている内部認証鍵と計算しその結果出力を端末が判断し、IC カードの正当性を確認する。サーバが IC カードの正当性を認証する為に利用できる。（注）

注 : 今回の署名機能の相互運用性確保の範囲では、利用シーケンスに含まない。利用した場合には、相互運用の妨げになる可能性があることに留意する必要がある。

第５章　相互運用性確保のための仕様

５．１　相互運用性確保

５．１．１　相互運用性を確保する範囲

本ガイドラインが目標とする相互運用性の確保は、異なる HPKI 認証局が発行した IC カードを用いて、医療従事者等の端末環境に依存せず電子署名を実現することにある。そのためには、どの HPKI 認証局が発行した IC カードに対しても、クライアントの電子署名アプリケーションが電子署名を実行可能とすることが必要になる。

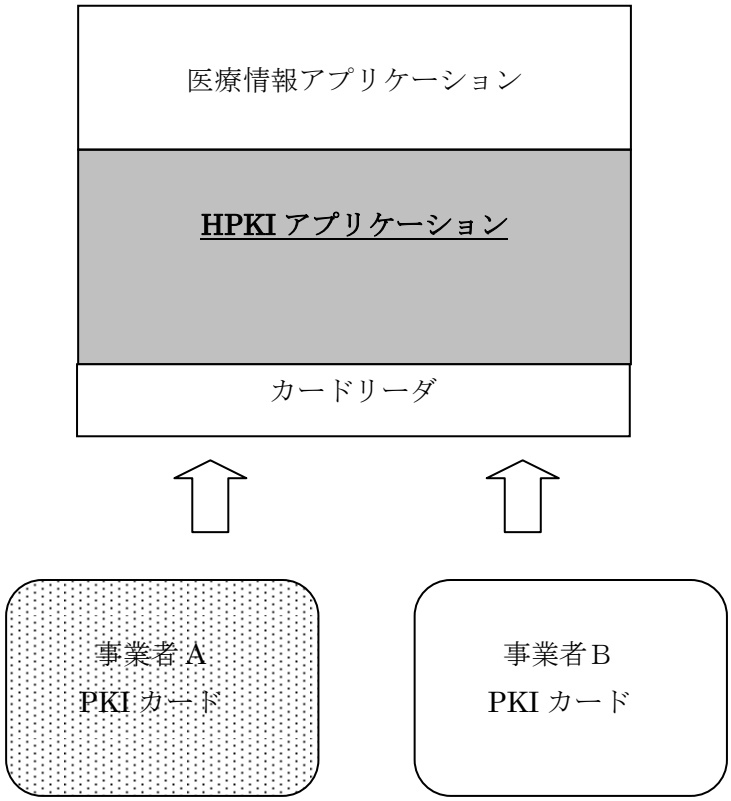

図１　相互運用の範囲

このような相互運用性を検討する場合、クライアント側のソフトウェアを以下の層に分けて検討する必要がある。

1) HPKI アプリケーション層
2) PKI 機能を提供する汎用ミドルウェア（PKCS #11 など医療アプリケーションと PKI アプリケーションの間のインタフェース）層
3) IC カードの入出力を提供する汎 IC カードリーダのドライバ層

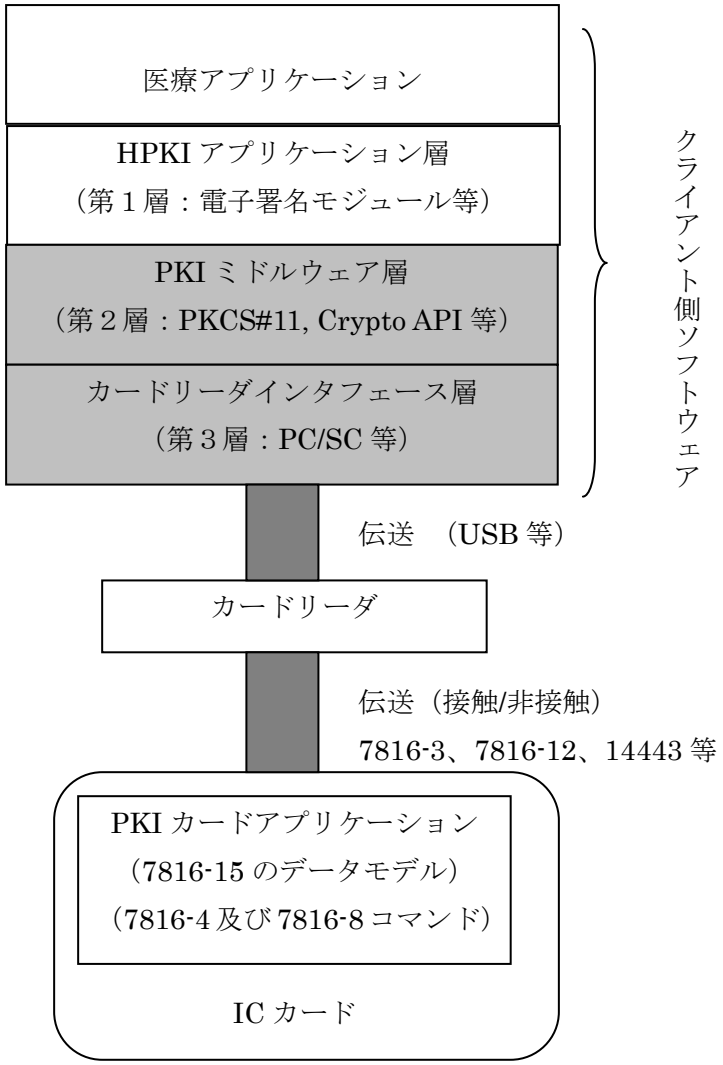

図２　PKI カードアプリケーションの基本構造

第１層は HPKI を利用する署名などの機能を提供する層で、本ガイドラインの対象外となる。HPKI サービスを提供することになる。

第２層は、HPKI に特化しない汎用的な PKI の機能を提供する層で、PKCS #11 あるいは Crypto API（CAPI）など、暗号機能を提供するモジュールが相当する。具体的には、IC カード内の PKI カードアプリケーションのデータモデルに従い、要求された PKI 機能を実現するカードコマンドに変換することによって IC カード内の PKI カードアプリケーションの機能を利用するための機能を提供する。PKI カードアプリケーションのデータモデルに関連する標準が ISO/IEC 7816-15 あるいは JIS X 6320-15 であり、カードコマンドに関連する標準が ISO/IEC 7816-4、ISO/IEC 7816-8 あるいは JIS X 6320-4、JIS X 6320-8 となる。第 2 層に相当するモジュールの機能とインタフェースの規定することによって、異なる HPKI 認証局が発行した IC カードによらず電子署名の機能を実現することが可能となり、相互運用性を確保したことになる。特に各 HPKI 認証局が提供する PKI カードアプリケー

ションの相互運用性を高めるためには、PKI カードアプリケーションのデータモデル、それを利用するためのカードコマンドの相互運用性を高める必要がある。これまでの多くのPKI 環境では、発行者、仕様の異なる IC カードによって同じ機能を提供する必要性がなかったため、この部分の相互運用性が考慮されておらず、問題を生じることがあった。
第 3 層は、汎用的な IC カードリーダとの入出力を実現する層で、第 2 層から受けた IC カードに対する命令（コマンド）を、IC カードリーダを通じて IC カードに伝える機能を提供する。プロトコルのハンドリング、IC カード及びカードリーダの制御などが含まれる。Windows では、PC/SC と呼ばれるモジュールがこの層に相当する。第 3 層のモジュールの機能とインタフェースの規定によって、対象となる IC カードとカードリーダの物理的な違い(例えば、接点付きと非接触など)によらない、汎用的な IC カード入出力を実現することができる。これによって、異なるインタフェースの IC カードや異なるカードリーダを利用した場合でも、IC カードと IC カード内の PKI カードアプリケーションに対して同じ入出力を実現することができる。
本報告書では、第 2 層のインタフェースを５．２に記述する。また、第 2 層で利用する IC カード内の PKI カードアプリケーションのデータモデルを５．３に、カードコマンドを６．４で説明する。

５．１．２　相互運用性を確保するための条件

本ガイドラインでは、異なる HPKI 認証局が発行した IC カードの PKI カードアプリケーションでの相互運用性確保のための仕様を以下の条件で示すものとする。

１）相互運用の範囲
IC カードは、発行、個人向け初期化、利用、変更、廃棄などのライフサイクルのいずれかの状態に属する。本ガイドラインの相互運用性は、利用の部分、特に電子署名を行う部分のみの相互運用を範囲とする。

２）証明書の適応範囲
HPKI 認証局が発行した証明書で、証明書の適応範囲は電子署名（non repudiation）とする。

３）複数の証明書保持
カード内には、エンドエンティティの証明書だけではなく、HPKI 認証局の証明書（複数の場合がある）、厚労省の HPKI ルート認証局の自己署名証明書を含めた証明書が格納できる。

４）複数の資格への対応
複数の資格を持つ医療従事者が存在し、業務によって証明書を使い分ける必要がある。本ガイドラインでは、１枚の証明書には１つの hcRole（保健医療福祉分野での役割、資格）が格納されているものとする。複数の資格を使い分ける際には、複数の PKI カードアプリケーションによって、解決するものとする。当面は、１枚の IC カードには１PKI カードアプリケーションであると想定するが、将来は複数の PKI カードアプリケーションが１枚の IC カードに搭載される可能性があることを前提とする。

５）ミドルウェアの HPKI 対応

HPKI の証明書には HC Role が記述されているが、ミドルウェア層では HC Role を用いた HPKI 独自の処理は行わない。必要な場合には、クライアントアプリケーションで行うものとする。

６）IC カードの利用環境

カードの利用環境は、「医療情報システムの安全管理に関するガイドライン」などに沿って管理された医療機関内の端末を想定する。悪意をもったソフトウェアなどが存在せず、安全に稼動するよう管理されているものとする。そのため、IC カードと利用する端末の間の認証、通信の安全性を確保するセキュアメッセージングなどは本ガイドラインの範囲外とする。

７）対象とする IC カード

利用されるカードの仕様は、特に限定しない。発行時の構成を変えない IC カード、発行後にアプリケーションを追加可能な IC カード等、いずれもが含まれるものとする。

８）標準準拠

長期に渡る相互運用性を確保するため、既存の JIS あるいは ISO に準拠した仕様とする。

９）私有鍵と証明書のアクセス権

電子署名の際には、毎回 PIN 入力によるカード保有者認証を行うものとする。PIN の変更は、相互運用性を確保する範疇には含めないものとする。証明書の読み出しは、アクセス制御を行わないものとする。

１０）　パスワード（PIN）の入力

アプリケーションの性質により、アプリケーション側で PIN を保持し、連続した署名実施の際にユーザ側の PIN 入力を省略することは許容する。

５．２　アプリケーションプログラムとのインタフェース

本ガイドラインでは、PKCS #11 インタフェースと、Crypto API インタフェースの２種類のインタフェースを想定する。それぞれが満たさなければならない最小限の機能を 5.2.1 及び 5.2.2 で説明する。

共通の条件としては

- マルチスレッドから 1 枚のカードへの同時アクセスは想定しない。複数のスレッドからアクセスする必要がある場合には、実行条件に注意する必要がある。
- 署名の際には、インタフェースレベルでは毎回 PIN を設定することを推奨する。但し、アプリケーション側で PIN を保持して利用することは妨げないので、アプリケーションの性質によって決定する必要がある。ISO/IEC 7816-15、最新の PKCS #11 の仕様には署名を行う際に毎回 PIN を要求する仕様が含まれており、新しい流れとして採用するところが増えている。そのため、将来は署名の際には毎回 PIN を要求する仕様が主流になると考えられる。
- PIN に設定されるパスワードの文字セットは、ASCII とする（CryptoAPI の制限によ

る）
となる。

５．２．１　PKCS #11 インタフェース

相互運用を実現するために必要となる、PKCS #11 のサポートすべき機能の一覧を示す。

表１　PKCS #11 の関数一覧

No	API 名	概要
1	C_GetFunctionList	関数ポインタリストを取得する。
2	C_Initialize	PKCS #11 ライブラリを初期化する。
3	C_Finalize	PKCS #11 ライブラリを終了する。
4	C_GetInfo	ライブラリ情報を取得する。
5	C_GetSlotList	スロットリストを取得する。
6	C_GetSlotInfo	スロット情報を取得する。
7	C_GetTokenInfo	トークン情報を取得する。
8	C_GetMechanismList	サポートメカニズム（アルゴリズム）を取得する。
9	C_GetMechanismInfo	メカニズム（アルゴリズム)情報を返す。
10	C_OpenSession	セッションを確立する。
11	C_CloseSession	セッションを切断する。
12	C_CloseAllSessions	すべてのセッションを切断する。
13	C_GetSessionInfo	セッション状態を取得する。
14	C_Login	トークンをログイン状態にする。
15	C_Logout	トークンをログアウト状態にする。
16	C_FindObjectsInit	オブジェクトの検索を開始する。
17	C_FindObjects	オブジェクトの検索を行う。
18	C_FindObjectsFinal	オブジェクトの検索を終了する。
19	C_GetAttributeValue	オブジェクトの属性値を取得する。
20	C_SignInit	署名処理を初期化する。
21	C_Sign	データに署名を行う。

サポートすべき API の仕様は以下の通りとなる。なお、戻り値および構造体に関しては、PKCS #11 を参照のこと。

(1) C_GetFunctionList

API 名	C_GetFunctionList
概要	関数ポインタリストを取得する。

関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetFunctionList) (　CK_FUNCTION_LIST_PTR_PTR　　ppFunctionList) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_FUNCTION_LIST_PTR_PTR	OUT	関数アドレスリストポインタ

(2) C_Initialize

API 名	C_Initialize		
概要	PKCS #11 ライブラリを初期化する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_Initialize) (　CK_VOID_PTR　　pInitArgs) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_VOID_PTR	IN	NULL ポインタを指定

(3) C_Finalize

API 名	C_Finalize		
概要	PKCS #11 ライブラリを終了する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_Finalize) (　CK_VOID_PTR　　pReserved) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_VOID_PTR	IN	NULL ポインタを指定

(4) C_GetInfo

API 名	C_GetInfo		
概要	ライブラリ情報を取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetInfo) (　CK_INFO_PTR　　pInfo) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_INFO_PTR	OUT	ライブラリ情報ポインタ

備考	取得可能なライブラリ情報は以下の通り。 CK_INFO::cryptokiVersion: PKCS #11 規格バージョン: 2.20 CK_INFO::manufacturerID: ライブラリ製造者名: CK_INFO::flags: ビットフラグ: 0 CK_INFO::libraryDescription: ライブラリ記述文:「HPKI 1.0」 CK_INFO::libraryVersion: ライブラリバージョン

(5) C_GetSlotList

API 名	C_GetSlotList		
概要	スロットリストを取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSlotList) (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_BBOOL	IN	TRUE: カード有りのスロットリストを返す FALSE: 接続されているすべてのスロットリストを返す
	CK_SLOT_ID_PTR	IN/OUT	スロット ID リストポインタ
	CK_ULONG_PTR	IN/OUT	スロット ID リスト件数

(6) C_GetSlotInfo

API 名	C_GetSlotInfo		
概要	スロット情報を取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSlotInfo) (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容

引数	CK_SLOT_ID	IN	スロット ID
	CK_SLOT_INFO_PTR	OUT	スロット情報ポインタ
備考	取得可能なスロット情報は以下の通り。 CK_SLOT_INFO::slotDescription: スロット記述文: PCSC リーダ名称 CK_SLOT_INFO::manufacturerID: スロット製造者名: なし(32 バイトの空白文字列) CK_SLOT_INFO::flags: ビットフラグ: カード有無 (カードがあれば CKF_TOKEN_PRESENT) CK_SLOT_INFO::hardwareVersion: スロットハードウェアバージョン: 0.0 CK_SLOT_INFO::firmwareVersion: スロットファームウェアバージョン: 0.0		

(7) C_GetTokenInfo

API 名	C_GetTokenInfo		
概要	トークン情報を取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetTokenInfo) (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_TOKEN_INFO_PTR	OUT	トークン情報ポインタ

備考	取得可能なトークン情報は以下の通り。 CK_TOKEN_INFO::label：ラベル名 ： 「HPKI Application」：カードにラベル有りの場合 なし(32 バイトの空白文字列)：カードにラベル無しの場合 CK_TOKEN_INFO::model：デバイスモデル名 ： 「ISO 7816-15:2004」または「JIS X 6320-15:2006」 CK_TOKEN_INFO::flags：ビットフラグ ： 乱数生成器有(CKF_RNG) | ユーザログイン要(CKF_LOGIN_REQUIRED) | ユーザ PIN の初期化有(CKF_USER_PIN_INITIALIZED) CK_TOKEN_INFO::ulMaxPinLen：PIN の最大長 ： 16 CK_TOKEN_INFO::ulMinPinLen：PIN の最小長 ： 4 値は、付属書Ｂの構造例を用いた場合の例である。

(8) C_GetMechanismList

API 名	C_GetMechanismList		
概要	サポートメカニズム(アルゴリズム)を取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismList) (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_MECHANISM_TYPE_PTR	IN/OUT	メカニズムタイプポインタ
	CK_ULONG_PTR	IN/OUT	メカニズムタイプ件数
備考	取得可能なサポートメカニズムは以下の通り。 Sign 用: CKM_RSA_PKCS		

(9) C_GetMechanismInfo

API 名	C_GetMechanismInfo
概要	メカニズム(アルゴリズム)情報を返す。

関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismInfo) (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_MECHANISM_TYPE	IN	メカニズムタイプ
	CK_MECHANISM_INFO_PTR	OUT	メカニズム情報ポインタ
備考	設定する情報は以下の通り。 type = CKM_RSA_PKCS の場合 CK_MECHANISM_INFO::ulMinKeySize： 最小鍵長: 1024 CK_MECHANISM_INFO::ulMaxKeySize： 最大鍵長: 1024(2048 も可) CK_MECHANISM_INFO::flags： ビットフラグ ： C_SignInit 利用可能(CKF_SIGN) 値は、付属書Ｂの構造例を用いた場合の例である。		

(10) C_OpenSession

API 名	C_OpenSession		
概要	セッションを確立する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_OpenSession) (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY Notify, CK_SESSION_HANDLE_PTR phSession) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_FLAGS	IN	CKF_SERIAL_SESSION を指定
	CK_VOID_PTR	IN	NULL ポインタを指定
	CK_NOTIFY	IN	NULL ポインタを指定
	CK_SESSION_HANDLE_PTR	OUT	セッションハンドルポインタ

(11) C_CloseSession

API 名	C_CloseSession		
概要	セッションを切断する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_CloseSession) (　CK_SESSION_HANDLE　　hSession) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

(12) C_CloseAllSessions

API 名	C_CloseAllSessions		
概要	すべてのセッションを切断する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_CloseAllSessions) (　CK_SLOT_ID　slotID) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID

(13) C_GetSessionInfo

API 名	C_GetSessionInfo		
概要	セッション状態を取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSessionInfo) (　CK_SESSION_HANDLE　　hSession, 　CK_SESSION_INFO_PTR　　pInfo) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_SESSION_INFO_PTR	OUT	セッション状態ポインタ
備考	以下の状態を返す。 　ログインしていないとき: CKS_RO_PUBLIC_SESSION 　　(CKS_RW_PUBLIC_SESSION でも可) 　ログインしているとき: CKS_RO_USER_FUNCTIONS 　　(CKS_RW_USER_FUNCTIONS でも可)		

(14) C_Login

API 名	C_Login		
概要	トークンをログイン状態にする。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_Login) (　CK_SESSION_HANDLE　　hSession, 　CK_USER_TYPE　　userType, 　CK_UTF8CHAR_PTR　　pPin, 　CK_ULONG　　ulPinLen) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_USER_TYPE	IN	CKU_USER を指定
	CK_UTF8CHAR_PTR	IN	パスワード文字列ポインタ。文字列は、ASCII とする。
	CK_ULONG	IN	パスワード文字列長

(15) C_Logout

API 名	C_Logout		
概要	トークンをログアウト状態にする。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_Logout) (　CK_SESSION_HANDLE　　hSession) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

(16) C_FindObjectsInit

API 名	C_FindObjectsInit		
概要	オブジェクトの検索を開始する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsInit) (　CK_SESSION_HANDLE　　hSession, 　CK_ATTRIBUTE_PTR　　pTemplate, 　CK_ULONG　　ulCount) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_ATTRIBUTE_PTR	IN	属性テーブルポインタ

	CK_ULONG	IN	属性テーブル数
備考	以下の属性による検索を行う。 CKA_CLASS　CKO_CERTIFICATE または CKO_PRIVATE_KEY CKA_TOKEN　True CKA_LABEL　証明書の名前 または 私有鍵の名前 CKA_ID　証明書の番号 または 私有鍵の番号 CKA_CERTIFICATE_TYPE　CKC_X_509 CKA_VALUE　証明書の値 CKA_KEY_TYPE　CKK_RSA CKA_MODULUS　利用者公開鍵の N CKA_PUBLIC_EXPONENT　利用者公開鍵の E		

(17) C_FindObjects

API 名	C_FindObjects		
概要	オブジェクトの検索を行う。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjects) (CK_SESSION_HANDLE　hSession, CK_OBJECT_HANDLE_PTR　phObject, CK_ULONG　ulMaxObjectCount, CK_ULONG_PTR　pulObjectCount) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE_PTR	OUT	オブジェクトハンドルポインタ
	CK_ULONG	IN	最大オブジェクト数
	CK_ULONG_PTR	OUT	オブジェクト数ポインタ

(18) C_FindObjectsFinal

API 名	C_FindObjectsFinal		
概要	オブジェクトの検索を終了する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsFinal) (CK_SESSION_HANDLE　hSession) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

(19) C_GetAttributeValue

API 名	C_GetAttributeValue		
概要	オブジェクトの属性値を取得する。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetAttributeValue) (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル
	CK_ATTRIBUTE_PTR	IN/OUT	属性テーブルポインタ
	CK_ULONG	IN	属性テーブル数
備考	以下の属性に対する値が取得可能である。 CKA_CLASS　CKO_CERTIFICATE または CKO_PRIVATE_KEY CKA_TOKEN　True CKA_PRIVATE　表２　オブジェクト情報一覧参照。 CKA_LABEL　表２　オブジェクト情報一覧参照。 CKA_ID　表２　オブジェクト情報一覧参照。 CKA_CERTIFICATE_TYPE　CKC_X_509 CKA_VALUE　表２　オブジェクト情報一覧参照。 CKA_ISSUER　表２　オブジェクト情報一覧参照。 CKA_SERIAL_NUMBER　表２　オブジェクト情報一覧参照。 CKA_SUBJECT　表２　オブジェクト情報一覧参照。 CKA_KEY_TYPE　CKK_RSA CKA_SIGN　True CKA_EXTRACTABLE　False CKA_ALWAYS_AUTHENTICATE　True CKA_MODULUS　表２　オブジェクト情報一覧参照。 CKA_PUBLIC_EXPONENT　表２　オブジェクト情報一覧参照。 値は、付属書Ｂの構造例を用いた場合の例である。		

表２ オブジェクト属性一覧

#	オブジェクト	属性名	属性値
1	利用者鍵	CKA_PRIVATE	True
		CKA_LABEL	Private key of HPKI

		CKA_ID	17H
		CKA_MODULUS	利用者証明書から取得した値
		CKA_PUBLIC EXPONENT	利用者証明書から取得した値
2	利用者証明書	CKA_PRIVATE	False
		CKA_LABEL	HPKI END ENTITY CERTIFICATE
		CKA_ID	17H
		CKA_VALUE	利用者証明書自体
		CKA_ISSUER	利用者証明書から取得した値
		CKA_SERIAL_NUMBER	利用者証明書から取得した値
		CKA_SUBJECT	利用者証明書から取得した値
3	厚労省ルート CA 証明書	CKA_PRIVATE	False
		CKA_LABEL	MHLW CA CERTIFICATE
		CKA_ID	19H
		CKA_VALUE	厚労省ルート CA 証明書自体
		CKA_ISSUER	厚労省ルート CA 証明書から取得した値
		CKA_SERIAL_NUMBER	厚労省ルート CA 証明書から取得した値
		CKA_SUBJECT	厚労省ルート CA 証明書から取得した値
4	日医、MEDIS ルート CA 証明書	CKA_PRIVATE	False
		CKA_LABEL	HPKI ROOT CA CERTIFICATE
		CKA_ID	1AH
		CKA_VALUE	日医、MEDIS ルート CA 証明書自体
		CKA_ISSUER	日医、MEDIS ルート CA 証明書から取得した値
		CKA_SERIAL_NUMBER	日医、MEDIS ルート CA 証明書から取得した値
		CKA_SUBJECT	日医、MEDIS ルート CA 証明書から取得した値
5	MEDIS 中間 CA 証明書	CKA_PRIVATE	False
		CKA_LABEL	HPKI CA CERTIFICATE
		CKA_ID	1BH
		CKA_VALUE	MEDIS 中間 CA 証明書自体
		CKA_ISSUER	MEDIS 中間 CA 証明書から取得した値
		CKA_SERIAL_NUMBER	MEDIS 中間 CA 証明書から取得した値
		CKA_SUBJECT	MEDIS 中間 CA 証明書から取得した値

注：　値は、付属書Ｂの構造例を用いた場合の例である。

(20) C_SignInit

API 名	C_SignInit
概要	署名処理を初期化する。

関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_SignInit) (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_RSA_PKCS のみ指定可
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル

(21) C_Sign

API 名	C_Sign		
概要	データに署名を行う。		
関数インタフェース	CK_DEFINE_FUNCTION(CK_RV, C_Sign) (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen) ;		
戻り値	CK_RV (CKR_OK: 成功 その他: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	データポインタ
	CK_ULONG	IN	データ長
	CK_BYTE_PTR	IN/OUT	署名データポインタ
	CK_ULONG_PTR	IN/OUT	署名データ長ポインタ

表３　構造体仕様

NO	構造体名	概要
1	CK_INFO CK_INFO_PTR	PKCS #11 ライブラリ情報
2	CK_SLOT_ID CK_SLOT_ID_PTR	スロット ID

3	CK_SLOT_INFO CK_SLOT_INFO_PTR	スロット情報
4	CK_TOKEN_INFO CK_TOKEN_INFO_PTR	トークン情報
5	CK_SESSION_HANDLE CK_SESSION_HANDLE_PTR	セッションハンドル
6	CK_USER_TYPE	ユーザタイプ
7	CK_SESSION_INFO CK_SESSION_INFO_PTR	セッション情報
8	CK_OBJECT_HANDLE CK_OBJECT_HANDLE_PTR	オブジェクトハンドル
9	CK_ATTRIBUTE CK_ATTRIBUTE_PTR	属性タイプ、値、長さを含む構造体
10	CK_MECHANISM_TYPE CK_MECHANISM_TYPE_PTR	メカニズムタイプ
11	CK_MECHANISM CK_MECHANISM_PTR	メカニズムタイプを含む、メカニズムを示す構造体
12	CK_MECHANISM_INFO CK_MECHANISM_INFO_PTR	メカニズム情報
13	CK_RV	ライブラリの戻り値
14	CK_NOTIFY	コールバック情報
15	CK_FUNCTION_LIST CK_FUNCTION_LIST_PTR CK_FUNCTION_LIST_PTR_PTR	PKCS #11 ライブラリのバージョン、関数ポインタを含む構造体

５．２．２　Crypto API インタフェース

本ガイドラインでは、Crypto API のサポートすべき機能の一覧を示す。

表４　Criytp API

No	API 名	概要
1	CryptAcquireContext	鍵コンテナのハンドルを生成する
2	CryptReleaseContext	鍵コンテナのハンドルを開放する。
3	CryptGetProvParam	CSP のパラメータの値を取得する。
4	CryptSetProvParam	CSP のパラメータの値を設定する。
5	CryptDestroyKey	鍵の破棄を行う。
6	CryptGetKeyParam	鍵のパラメータの値を取得する。

7	CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する。
8	CryptCreateHash	ハッシュオブジェクトの生成を行う。
9	CryptDestroyHash	ハッシュオブジェクトの破棄を行う。
10	CryptSetHashParam	ハッシュオブジェクトのパラメータを設定する。
11	CryptSignHash	ハッシュ値に署名を行う。

サポートすべき API の仕様は以下の通りとなる。なお、戻り値、構造体は、CryptoAPI の仕様を参照のこと。

(1) CryptAcaquireContext

API 名	CryptAcquireContext		
概要	鍵コンテナのハンドルを生成する		
関数インタフェース	CryptAcquireContext(HCRYPTPROV* phProv, LPCTSTR pszContainer, LPCTSTR pszProvider, DWORD dwProvType, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTPROV*	OUT	ハンドル格納場所のポインタ
	LPCTSTR	IN	NULL 文字で終了するコンテナ名称 NULL を指定すること。
	LPCTSTR	IN	NULL 文字で終了する CSP 名称 ” HPKI 1.0”を指定すること。
	DWORD	IN	DWORD IN プロバイダタイプ PROV_RSA_FULL あるいは PROV_RSA_SIG を指定すること。
	DWORD	IN	動作に関するパラメータ 0: 利用者証明書、利用者私有鍵使用時に指定する。
備考	dwFlags に”0”を指定した場合、パスワード入力画面が表示される。 CRYPT_SILENT を設定した場合ユーザインタフェースは表示されないので、CryptSetProvParam で PP_SIGNATURE_PIN を指定して PIN の情報を渡す必要がある。		

(2) CryptReleaseContext

API 名	CryptReleaseContext		
概要	鍵コンテナのハンドルを開放する		
関数インタフェース	CryptAcquireContext(HCRYPTPROV* phProv, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTPROV*	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	動作に関するパラメータ 0 を指定すること。

(3) CryptGetProvParam

API 名	CryptGetProvParamt		
概要	CSP のパラメータの値を取得する。		
関数インタフェース	CryptGetProvParam(HCRYPTPROV hProv, DWORD dwParam, BYTE* pbData, DWORD* pdwDataLen, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTPROV*	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	値を取得するためのパラメータサポートする値を表 Y に示す
	BYTE *	OUT	値を格納するバッファのポインタ NULL を指定した場合は値の読み込みは行われず、pdwDataLen に値の格納に必要な長さが設定される。

	DWORD*	IN/OUT	値の長さを保持するバッファへのポインタ 関数呼び出し時には、pbDataバッファに割り当てられたメモリサイズを指定する。関数終了時には、値の格納に必要な長さが設定される。
	DWORD	IN	動作に関するパラメータ。CSP が管理する鍵コンテナのうちの最初のコンテナを指定する場合には、PP_ENUMCONTAINERS | PP_CRYPT_FIRST を、それ以降の鍵コンテナを指定する場合には、PP_ENUMCONTAINERS を指定する。 その他の利用では、動作に関するパラメータ 0 を指定すること。

表５　CryptGetProvParam の dwParam でサポートする値

#	値	内容
1	PP_IMPTYPE	CSP の実装形を DWORD 型で返す。
2	PP_NAME	CSP 名称を CHAR 型の NULL ターミネート文字列で返す。本 CSP は、”HPKI Crypto Service Provider”を返す。
3	PP_VERSION	CSP のバージョンを DWORD 型で返す。
4	PP_PROVTYPE	CSP のプロバイダタイプを DWORD 型で返す。
5	PP_CERTCHAIN	CSP に含まれる CA の証明書（チェイン）を取得する。

注：CA の証明書のチェインの順番は特に規定しない。

CA のフォーマットは、ASN.1 でエンコードされた

```
SET OF Certificate ; -- X.509 certificates
```

に従うものとする。

(4) CryptSetProvParam

API 名	CryptSetProvParamt
概要	CSP のパラメータの値を設定する。

関数インタフェース	CryptGetProvParam(HCRYPTPROV hProv, DWORD dwParam, BYTE* pbData, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTPROV*	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	値を取得するためのパラメータサポートする値を表 Y に示す
	BYTE *	IN	値を格納するバッファのポインタ。呼ぶ前に値を設定する。
	DWORD	IN	動作に関するパラメータ 0 を指定すること。

表６　CryptSetProvParam の dwParam でサポートする値

#	値	内容
1	PP_SIGNATUER_PIN	CSP を署名で利用するための PIN を設定する。pbData には、NULL で終わる ASCII 文字列を渡す。

CryptAquireContext で CRYPT_SILENT を設定した場合、PP_SIGNATURE_PIN を指定して PIN の情報を渡す必要がある。指定しない場合には、CSP 内のユーザインタフェースによって PIN の入力が促される。

(5) CryptDestroyKey

API 名	CryptDestroyKey		
概要	鍵の破棄を行う。		
関数インタフェース	CryptDestroyKey(HCRYPTKEY hKey);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTKEY	IN	破棄する鍵のハンドル

(6) CryptGetKeyParam

API 名	CryptGetKeyParam		
概要	鍵のパラメータの値を取得する。 (IC カードに格納された利用者証明書(DER 形式)を返す。)		
関数インタフェース	CryptGetKeyParam(HCRYPTKEY hKey, DWORD dwParam, BYTE* pbData, DWORD* pdwDataLen, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功 FALSE:失敗）) GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTKEY	IN	値を取得する鍵のハンドル
	DWORD	IN	値を取得するパラメータ KP_CERTIFICATE: IC カードに格納された利用者証明書を返す。
	DWORD*	IN/OUT	値の長さを保持するバッファへのポインタ 関数呼び出し時には、pbDataバッファに割り当てられたメモリサイズを指定する。関数終了時には、値の格納に必要な長さが設定される。
	DWORD	IN	動作に関するパラメータ 0 を指定すること。

(7) CryptGetUserKey

API 名	CryptGetUserKey
概要	鍵コンテナ内の鍵ハンドルを取得する
関数インタフェース	CryptGetUserKey(HCRYPTPROV hProv, DWORD dwKeySpec, HCRYPTKEY* phUserKey);
戻り値	BOOL（TRUE:成功 FALSE:失敗）) GetLastError にてエラー情報が取得できる。

	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	鍵の種類 AT_SIGNATURE: 署名用鍵を設定すること。
	HCRYPTKEY*	OUT	鍵ハンドルをコピーするバッファのアドレス

(8) CryptCreateHash

API 名	CryptCreateHash		
概要	ハッシュオブジェクトの生成を行う。		
関数インタフェース	CryptCreateHash(HCRYPTPROV hProv, ALG_ID Algid, HCRYPTKEY hKey, DWORD dwFlags, HCRYPTHASH* phHash);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	ALG_ID	IN	CALG_SHA1: SHA1 アルゴリズム 現在公開されている仕様では、対象範囲が不明確であるが、将来以下の SHA2 ファミリが利用されることに注意が必要である。 CALG_SHA_256 CALG_SHA_384 CALG_SHA_512
	HCRYPTKEY	IN	キードハッシュの場合の鍵ハンドル 0 を設定すること。(本 CSP ではキードハッシュはサポートしない
	DWORD	IN	動作に関するパラメータ 0 を設定すること。

	HCRYPTHASH*	OUT	生成したハッシュオブジェクトのハンドルをコピーするバッファのポインタ

(9) CryptDestroyHash

API 名	CryptDestroyHash		
概要	ハッシュオブジェクトの破棄を行う。		
関数インタフェース	CryptDestroyHash(HCRYPTHASH hHash);		
戻り値	BOOL（TRUE:成功 FALSE:失敗）） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTHASH	IN	破棄するハッシュオブジェクトのハンドル

(10) CryptSetHashParam

API 名	CryptSetHashParam		
概要	ハッシュオブジェクトのパラメータを設定する。		
関数インタフェース	CryptSetHashParam(HCRYPTHASH hHash, DWORD dwParam, BYTE* pbData, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功 FALSE:失敗）） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTHASH	IN	パラメータを設定するハッシュオブジェクトのハンドル
	DWORD	IN	設定するパラメータ HP_HASHVAL: pbData バッファに格納された値をハッシュ値としてハッシュオブジェクトに設定する。
	BYTE*	IN	パラメータに設定するデータのポインタ
	DWORD	IN	動作に関するパラメータ 0 を設定すること。

(11)　CryptSignHash

API 名	CryptSignHash		
概要	ハッシュ値に署名を行う。		
関数インタフェース	CryptSignHash(HCRYPTHASH hHash, DWORD dwKeySpec, LPCTSTR sDescription, DWORD dwFlags, BYTE* pbSignature, DWORD* pdwSigLen);		
戻り値	BOOL（TRUE:成功 FALSE:失敗） GetLastError にてエラー情報が取得できる。		
	型	I/O	内容
引数	HCRYPTHASH	IN	署名を行うハッシュオブジェクトのハンドル
	DWORD	IN	鍵の種類 署名用鍵 AT_SIGNATURE を設定すること。
	LPCTSTR	IN	ハッシュの概要についての NULL ターミネート文字列 NULL を設定すること
	DWORD	IN	署名時のフラグ 0 を設定すること。
	BYTE*	OUT	署名データを格納するバッファのポインタ NULL を指定した場合は値の書き込みは行われず、pdwSigLen に値の格納に必要な長さが設定される。
	DWORD*	IN/OUT	署名データの長さを保持するバッファへのポインタ 関数呼び出し時には、pbSignature バッファに割り当てられたメモリサイズを指定する。関数終了時には、署名データの格納に必要な長さが設定される。

５．３　PKI アプリケーションの構造

PKI カードアプリケーションのデータ構造は、JIS X 6320-15 (ISO/IEC 7816-15)で規定されている。本ガイドラインでは、この仕様を採用することとし、その概要をここで説明す

る。具体的なアプリケーションの構造を、付属書 B に示す。
JIS X 6320-15 は ISO/IEC 7816-15 を翻訳して JIS にしたもので、その内容は一致する。
JIS X 6320-15 は、IC カードの中に暗号演算を行うための鍵や、証明書を格納する際の仕様を定めている。RSA セキュリティ社が暗号機能の標準として検討した PKCS #15 をベースとしており、カード内にある暗号情報オブジェクトへのディレクトリ情報の記述方法を規定している。つまりどのような暗号情報オブジェクトが、カード内のどこにあり、どのようなアクセス条件になっているのかを記述できる仕様となっている。基本的な構造を図３に示す。

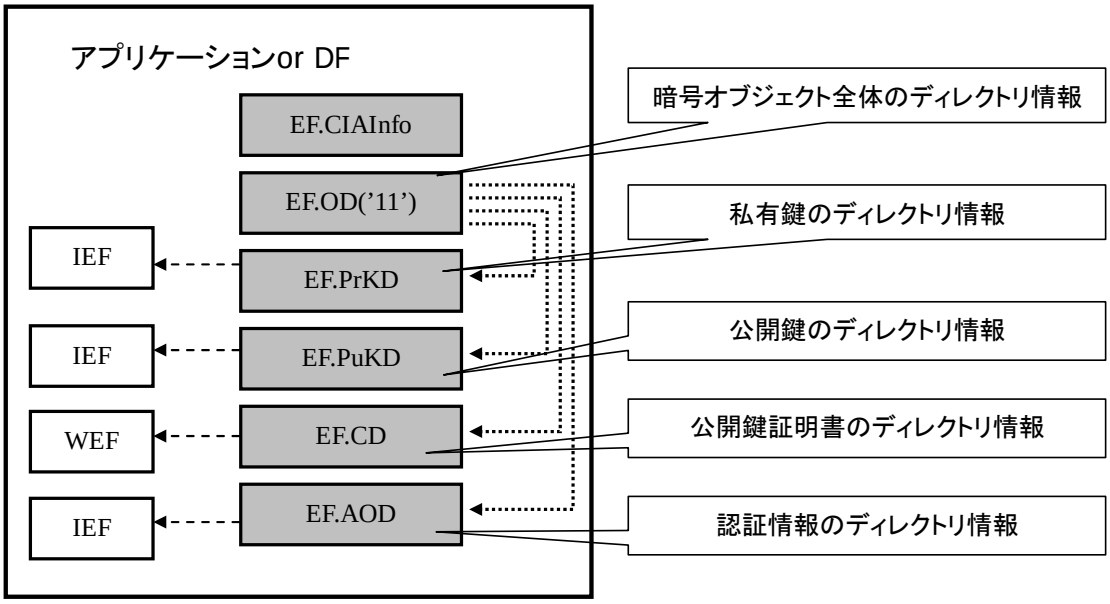

図３　JIS X 6320-15 の基本構成

暗号情報のディレクトリ情報を記述したファイル EF.OD は、各暗号情報オブジェクトのディレクトリ情報へのリンク（参照情報）を保持している。その情報から取得した各暗号情報オブジェクトのディレクトリ情報を取得し、最終的な暗号情報オブジェクトの所在、利用条件（セキュリティ属性）などを取得し、暗号情報オブジェクトを利用することになる。

５．４　PKI アプリケーションのコマンド仕様

５．３で説明した ISO/IEC 7816-15 に基づくデータモデル内の各暗号情報オブジェクト(鍵、証明書等）を利用するためのカードコマンドは、ISO/IEC 7816-4（JIS X 6320-4 と等しいが、JIS は出版前）, JIS X 6320-8 (ISO/IEC 7816-8 と等しい)によって規定される。本節では、そのコマンドの概要について説明する。

５．４．１　対象とするコマンド

電子署名に必要となる演算の実行に関連する IC カードに対するコマンドは、ISO/IEC 7816-4 及び ISO/IEC 7816-8 によって規定されている。表７に必要となるコマンドの一覧を示す。各コマンドの詳細については、付属書 C に記述する。

表７　相互運用性確保に必要となるコマンド

コマンド	規定される標準
SELECT	ISO/IEC 7816-4 (JIS X 6320-4)
VERIFY	ISO/IEC 7816-4 (JIS X 6320-4)
READ BINARY	ISO/IEC 7816-4 (JIS X 6320-4)
MANAGE SECURITY ENVIRONMENT	ISO/IEC 7816-4 (JIS X 6320-4)
PERFORM SECURITY OPERATION	ISO/IEC 7816-8 (JIS X 6320-8)

５．４．２　コマンドに対する制限

各コマンド共通の制限として以下を挙げる

・論理チャネルは 0 チャネル(基本チャネル)のみを使用する。

・セキュアメッセージングは使用しない

附属書 A （参考）PKI アプリケーション利用のシーケンス

署名のシーケンスは、ISO/IEC 7816-8(JIS X 6320-8)の附属書 A のシーケンスに従うものとする。本付属書では、A-1 にカードレベルでの利用シーケンスを、A-２に PKCS #11 インタフェースの利用シーケンスを、A-3 に CryptoAPI の利用シーケンスを示す。

Ａ－１　ＩＣカードの利用シーケンス

IC カードに対するコマンドレベルでの手順は、以下のシーケンスに従う。

Ａ－１－１　アプリケーション選択

１）アプリケーション/DF の選択

SELECT FILE

・“E8 28 BD 08 0F” をパーシャル DF 名として指定して選択する。パーシャル DF 名を用いた選択では、実際の DF 名の最初からのいくつか文字と一致する DF を検索し、そのレスポンスに示される DF 名が当初の目的とする DF 名であることを確認することで目的の DF が選択されたことになる。

・DF 名（AID）を FCI から取得する（サービス提供者の識別）

（１）コマンド

CLA	“00“
INS	“A4“ = SELECT
P1	“04“ = DF 名選択
P2	“00“ = 最初の DF 選択
Lc	“05“ = データフィールドの長さ
データ	“E8 28 BD 08 0F“ = ISO/IEC 7816-15 の AID （注）
Le	“00“ = FCI を受け取る

注 : 他の PKI アプリケーションと識別するため、HPKI 認証局の RID が追加される可能性がある

（２）レスポンス

	長さ	意味
データ	5～20	FCI(TAG=“84“) “6F” L1 “84“ L2　xx xx xx xx xx xx xx (xx は最長１６バイト)
SW1	1	
SW2	1	

（３）コマンド

CLA	“00“

INS	“A4“ = SELECT
P1	“04“ = DF 名選択
P2	“02“ = 次の DF 選択
Lc	“05“ = データフィールドの長さ
データ	“E8 28 BD 08 0F“ = ISO/IEC 7816-15 の AID
Le	“00“ = FCI を受け取る

（4）レスポンス

	長さ	意味
データ	5～20	FCP(TAG=“84“) “6F“ L1 “84” L2　xx xx xx xx xx xx xx (xx は最長１６バイト)
SW1	1	
SW2	1	

注）レスポンスの SW1 及び SW2 が’90 00’以外の場合のデータ長は、0 となる

（3）及び（4）を該当する DF が見つかるまで繰り返すが、該当する DF がなくなるとレスポンスとして

SW1	1	“6A”
SW2	1	“82”　（アクセス対象ファイルがない）

が返される。

A－１－２　証明書の読み出し

注：IC カード内の特定の HPKI アプリケーションの証明書を取得する。

１）アプリケーション/DF の選択

SELECT FILE

・A－１－１によって事前にアプリケーションを確認しており、選択する DF は事前にわかっているものとする。

（1）コマンド

CLA	“00“
INS	“A4“ = SELECT
P1	“04“ = DF 名選択
P2	“00“ = DF 選択
Lc	“XX“ = データフィールドの長さ(最長１６バイト)
データ	“E8 28 BD 08 0F XX XX XX XX“
Le	“00“ = FCI を受け取る

（2）レスポンス

	長さ	意味

データ	5～20	FC I (TAG=“84“)テンプレート
SW1	1	
SW2	1	

2）EF.CIAInfo の読み出し

READ BINARY (EF-ID=“12“)

（1）コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“92“ = 短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（2）レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

3）EF.OD の読み出し

READ BINARY (EF-ID=“11“)

（1）コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“91“ = 短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（2）レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

4）EF.CD 読み出し

READ BINARY (EF-ID=“15“)

（1）コマンド

CLA	“00“
INS	“B0“ = READ BINARY

P1	“95“ =短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（2） レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

5）証明書読み出し

READ BINARY (EF-ID=“18“)

（1） コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“98“ = 短縮 EF 指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（2） レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

ファイルの終わりに達していない場合には、READ BINARY のコマンドを続けて発行し、ファイルの終わりまで読み出す。

（3） コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“01“ = カレント EF 指定、オフセット（１５ビットの上位）
P2	“00“ = オフセット（１５ビットの下位８ビット）
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（4） レスポンス

	長さ	意味
データ	可変	データ
SW1	1	
SW2	1	

（以降ファイルの終わりに達するまで、オフセットを変えながら READ BINARY

コマンドを続けて発行する）

READ BINARY (EF-ID=“19“)

（1）コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“99“ = 短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（2）レスポンス

	長さ	意味
データ	可変	データ
SW1	1	
SW2	1	

（以降ファイルの終わりに達するまで、オフセットを変えながら READ BINARY コマンドを続けて発行する）

READ BINARY (EF-ID=“1A“)

（1）コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“9A“ =短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

2）レスポンス

	長さ	意味
データ	可変	データ
SW1	1	
SW2	1	

（以降ファイルの終わりに達するまで、オフセットを変えながら READ BINARY コマンドを続けて発行する）

(READ BINARY (EF-ID=“1B“))

（1）コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“9B“ = 短縮 EF 識別子指定

P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

２）レスポンス

	長さ	意味
データ	可変	データ
SW1	1	
SW2	1	

（以降ファイルの終わりに達するまで、オフセットを変えながら READ BINARY コマンドを続けて発行する）

A－１－３　署名計算

注：IC カード内の特定の HPKI アプリケーションの私有鍵によって署名演算する。事前に証明書が確認されて、DF が既知であることが必要。

１）アプリケーション/DF の選択

SELECT FILE

・選択する DF は事前にわかっているものとする。（証明書を確認して対象となる DF は既知である）。

（１）コマンド

CLA	“00“
INS	“A4“ = SELECT
P1	“04“ = DF 名選択
P2	“00“ = DF 選択
Lc	XX = データフィールドの長さ(最長１６バイト)
データ	“E8 28 BD 08 0F XX XX XX XX XX XX“
Le	“00“ = FCI を受け取る

（２）レスポンス

	長さ	意味
データ	5～20	FCI(TAG=’84’)テンプレート
SW1	1	
SW2	1	

２）EF.CIAInfo の読み出し

READ BINARY(EF-ID=“12“)

（１）コマンド

CLA	“00“
INS	“B0“ = READ BINARY

P1	“92“ = 短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

（2） レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

3） EF.OD の読み出し

READ BINARY (EF-ID=“11“)

（1） コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“91“ = 短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

2） レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

4） EF.AOD の読み出し

READ BINARY (EF-ID=“13“)

（1） コマンド

CLA	“00“
INS	“B0“ = READ BINARY
P1	“93“ = 短縮 EF 識別子指定
P2	“00“ = オフセット
Le	“00“ = 読み出しバイト数(256 バイト以内のファイルの終りまで)

2） レスポンス

	長さ	意味
データ	1～256	データ
SW1	1	
SW2	1	

5） EF.PrKD 読み出し

READ BINARY (EF-ID="14")

（1）コマンド

CLA	"00"
INS	"B0" = READ BINARY
P1	"94" = 短縮 EF 識別子指定
P2	"00" = オフセット
Le	"00" = 読み出しバイト数(256 バイト以内のファイルの終りまで)

2）レスポンス

	長さ	意味
データ	1～255	データ
SW1	1	
SW2	1	

6）署名

VERIFY (EF-ID="16") （毎回）

（1）コマンド

CLA	"00"
INS	"20" = VERIFY
P1	"00"
P2	"96" = 短縮 EF 識別子指定
Lc	データ長
	データ（PIN）

2）レスポンス

	長さ	意味
SW1	1	
SW2	1	

MANAGE SECURITY ENVIRONMENT

SET DST (P1P2="41 B6") ＋データ（鍵へのリファレンス: EF-ID="0017"）

（1）コマンド

CLA	"00"
INS	"22" = MANAGE SECURITY ENVIRONMENT
P1	"41" = 計算
P2	"B6" = 署名生成
Lc	"04" = データフィールドの長さ
データ	"81 02 00 17" = 署名生成鍵の IEF 指定

2）レスポンス

	長さ	意味
SW1	1	
SW2	1	

PERFORM SECURITY OPERATION

P1P2=“9E 9A“＋データ（パディングしたハッシュ値）

（1）コマンド

CLA	“00“
INS	“2A“ = PERFORM SECURITY OPERATION
P1	“9E“ = 署名計算
P2	“9A“ = データフィールドのデータに署名
Lc	XX = データフィールドの長さ
データ	署名するハッシュデータ

2）レスポンス

	長さ	意味
データ	XX	署名結果
SW1	1	
SW2	1	

A－2　PKCS #11 利用のシーケンス

(A) 初期処理

C_GetFunctionList	関数ポインタリストの取得 ppFunctionList：API アドレスポインタ

↓

C_Initialize	PKCS #11 ライブラリの初期化 pInitArgs：NULL_PTR

(B) 終了処理

C_Finalize	PKCS #11 ライブラリの終了処理 pReserved：NULL_PTR

(C) 証明書取得

初期化処理	(1)初期処理参照

↓

C_GetSlotList	スロットリスト数の取得 tokenPresent：CK_TRUE pSlotList：NULL_PTR pulCount：スロット数格納アドレス

IC カード内の HPKI アプリケーションの数を取得する必要がある。そのためには、
“E8 28 BD 08 0F”をパーシャル DF 名として指定して選択。戻り値が“6A 82”（アクセス対象ファイルがない）になるまで繰り返してアプリケーションの数を取得する。トークンの番号と、取得したアプリケーションの DF 名を対応させても良い。

↓

C_GetSlotList	スロットリストの取得 tokenPresent：CK_TRUE pSlotList：スロットリスト格納アドレス pulCount：スロット数格納アドレス

↓

ループ	①(pulCount-1)回分ループ (カウンタを i、初期値を 0 とする)

↓

C_GetSlotInfo	スロット情報の取得 slotID：C_GetSlotList で取得したスロットリストの i 番目(pSlotList[i]) pInfo：スロット情報格納アドレス

↓

C_GetTokenInfo	トークン情報の取得 slotID：C_GetSlotList で取得したスロットリストの i 番目(pSlotList[i]) pInfo：トークン情報格納アドレス

↓

C_OpenSession	アプリケーションとトークン間のセッションの確立 slotID：C_GetSlotList で取得したスロットリストの i 番目(pSlotList[i]) flags：CKF_SERIAL_SESSION pApplication：NULL_PTR Notify：NULL_PTR phSession：セッションハンドル格納アドレス

Ａ－１－２のシーケンスに従い、IC カード内のアプリケーションから全ての証明書を読み出す。

↓

C_FindObjectsInit	証明書検索操作の初期設定 hSession：C_OpenSession で取得したハンドル pTemplate：以下の属性を指定

	(1) type=CKA_CLASS value=CKO_CERTIFICATE (2) type=CKA_TOKEN value=CK_TRUE ulCount：設定する属性数(2)

↓

C_FindObjects	証明書の検索 hSession：C_OpenSession で取得したハンドル phObject：オブジェクトハンドル格納アドレス ulMaxObjectCount：オブジェクトハンドル格納領域数(4) pulObjectCount：発見したオブジェクト数格納アドレス

↓

ループ	②(pulObjectCount-1)回分ループ (カウンタを j、初期値を 0 とする)

↓

C_GetAttributeValue	証明書サイズの取得 hSession：C_OpenSession で取得したハンドル hObject：C_FindObjects で取得したオブジェクトハンドルリストの j 番目(phObject[j]) pTemplate：以下の属性を指定 (1) type=CKA_LABEL(※1) value=NULL_PTR ulValueLen=サイズ格納領域アドレス (2) type=CKA_VALUE(※2) value=NULL_PTR ulValueLen=サイズ格納領域アドレス ulCount：設定する属性数(2)

↓

C_GetAttributeValue	証明書の取得 hSession：C_OpenSession で取得したハンドル hObject：C_FindObjects で取得したオブジェクトハンドルリストの j 番目(phObject[j]) pTemplate：以下の属性を指定 (1) type=CKA_LABEL(※1) value=ラベル格納アドレス ulValueLen=格納領域サイズ (2) type=CKA_VALUE(※2) value=証明書格納アドレス

	ulValueLen=格納領域サイズ ulCount：設定する属性数(2)

↓

証明書のチェック	ラベル(※1)または証明書(※2)を用いて、求める証明書であるかを判定する。 求める証明書と一致すれば、次へ進む。 求める証明書と一致しなければ、カウンタ j を 1 加算して②に戻る。

↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession：C_OpenSession で取得したハンドル

↓

C_CloseSession もしくは、 C_CloseAllSessions	セッションの切断 C_CloseSession の場合 hSession：C_OpenSession で取得したハンドル C_CloseAllSessions の場合 slotID：C_OpenSession に指定したスロット ID

↓

証明書取得有無のチェック	求める証明書が取得できているかを判定する。 取得できていれば、次へ進む。 取得できていなければ、カウンタ i を 1 加算して①に戻る。

↓

終了処理	(2)終了処理参照

(D) 署名生成処理

署名に必要な私有鍵は、n(modulus)と e(publicExponent)から検索する。そのため、事前にすべての証明書を検索し、ラベルまたは証明書データによって求めている証明書か否かを判定しておく必要がある。

初期化処理	(1)初期処理参照

↓

C_GetSlotList	スロットリスト数の取得 tokenPresent：CK_TRUE pSlotList：NULL_PTR pulCount：スロット数格納アドレス

IC カード内の HPKI アプリケーションの数を取得する必要がある。そのためには、
“E8 28 BD 08 0F”をパーシャル DF 名として指定して選択。戻り値が“6A 82”（アクセス対象ファイルがない）になるまで繰り返してアプリケーションの数を取得する。トークンの番号と、取得したアプリケーションの DF 名を対応させても良い。

↓

C_GetSlotList	スロットリストの取得 tokenPresent：CK_TRUE pSlotList：スロットリスト格納アドレス pulCount：スロット数格納アドレス

↓

ループ	①(pulCount-1)回分ループ (カウンタを i、初期値を 0 とする)

↓

C_GetSlotInfo	スロット情報の取得 slotID：C_GetSlotList で取得したスロットリストの i 番目(pSlotList[i]) pInfo：スロット情報格納アドレス

↓

C_GetTokenInfo	トークン情報の取得 slotID：C_GetSlotList で取得したスロットリストの i 番目(pSlotList[i]) pInfo：トークン情報格納アドレス

↓

C_OpenSession	アプリケーションとトークン間のセッションの確立 slotID：C_GetSlotList で取得したスロットリストの i 番目(pSlotList[i]) flags：CKF_SERIAL_SESSION pApplication：NULL_PTR Notify：NULL_PTR phSession：セッションハンドル格納アドレス

↓

C_FindObjectsInit	証明書検索操作の初期設定 hSession：C_OpenSession で取得したハンドル pTemplate：以下の属性を指定 (1) type=CKA_CLASS value=CKO_CERTIFICATE (2) type=CKA_TOKEN value=CK_TRUE ulCount：設定する属性数(2)

Ａ－１－２のシーケンスに従い、IC カード内のアプリケーションから全ての証明書を読み出す。

↓

C_FindObjects	証明書の検索 hSession：C_OpenSession で取得したハンドル phObject：オブジェクトハンドル格納アドレス ulMaxObjectCount：オブジェクトハンドル格納領域数(4) pulObjectCount：発見したオブジェクト数格納アドレス

↓

ループ	②(pulObjectCount-1)回分ループ (カウンタを j、初期値を 0 とする)

↓

C_GetAttributeValue	証明書サイズの取得 hSession：C_OpenSession で取得したハンドル hObject：C_FindObjects で取得したオブジェクトハンドルリストの j 番目(phObject[j]) pTemplate：以下の属性を指定 (1) type=CKA_LABEL(※1) value=NULL_PTR ulValueLen=サイズ格納領域アドレス (2) type=CKA_VALUE(※2) value=NULL_PTR ulValueLen=サイズ格納領域アドレス ulCount：設定する属性数(2)

↓

C_GetAttributeValue	証明書の取得 hSession：C_OpenSession で取得したハンドル hObject：C_FindObjects で取得したオブジェクトハンドルリストの j 番目(phObject[j]) pTemplate：以下の属性を指定 (1) type=CKA_LABEL(※1) value=ラベル格納アドレス ulValueLen=格納領域サイズ (2) type=CKA_VALUE(※2) value=証明書格納アドレス ulValueLen=格納領域サイズ ulCount：設定する属性数(2)

↓

証明書のチェック	ラベル(※1)または証明書(※2)を用いて、求める証明書であるかを判定する。 求める証明書と一致すれば、次へ進む。

	求める証明書と一致しなければ、カウンタ j を 1 加算して②に戻る。

↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession：C_OpenSession で取得したハンドル

↓

証明書取得有無のチェック	求める証明書が取得できているかを判定する。 取得できていれば、次へ進む。 取得できていなければ、⑦の後、カウンタ i を 1 加算して①に戻る。

↓

E と N の取得	エンドエンティティ証明書から公開鍵の E と N を取得

↓

C_Login	トークンへのログイン hSession：C_OpenSession で取得したハンドル userType：CKU_USER pPin：パスワード文字列 ulPinLen：パスワード文字列長

A－1－3 の PIN 照合を行う。PIN 照合は、署名計算直前に必要となる場合がある。

↓

C_FindObjectsInit	私有鍵検索操作の初期設定 hSession：C_OpenSession で取得したハンドル pTemplate：以下の属性を指定 (1) type=CKA_CLASS value=CKO_PRIVATE_KEY (2) type=CKA_TOKEN value=CK_TRUE (3) type=CKA_MODULUS value=公開鍵の N (4) type=CKA_PUBLIC_EXPONENT value=公開鍵の E ulCount：設定する属性数(4)

↓

C_FindObjects	私有鍵の検索 hSession：C_OpenSession で取得したハンドル phObject：オブジェクトハンドル格納アドレス ulMaxObjectCount：オブジェクトハンドル格納領域数(1) pulObjectCount：発見したオブジェクト数格納アドレス

↓

C_FindObjectsFinal	私有鍵検索操作の終了処理

	hSession：C_OpenSession で取得したハンドル

↓

ダイジェスト値の生成	署名対象データのダイジェスト値を生成

↓

DigestInfo の作成	ダイジェスト値を使用して DigestInfo を作成

↓

C_SignInit	③署名処理の初期設定 hSession：C_OpenSession で取得したハンドル pMechanism：CKM_RSA_PKCS hKey：C_FindObjects で取得したオブジェクトハンドル

↓

C_Sign	④署名値長取得 hSession：C_OpenSession で取得したハンドル pData：DigestInfo ulDataLen：DigestInfo のデータ長 pSignature：NULL_PTR pulSignatureLen：署名値長格納アドレス

↓

C_Sign	⑤署名 hSession：C_OpenSession で取得したハンドル pData：DigestInfo ulDataLen：DigestInfo のデータ長 pSignature：署名値格納アドレス pulSignatureLen：署名値長格納アドレス

ここで、A－1－3 のハッシュの設定、署名計算の演算を実行する必要がある。PIN の照合はここで行っても良い。

↓

C_Logout	トークンからのログアウト hSession：C_OpenSession で取得したハンドル

↓

(連続して署名を行う場合)	(1 回の署名毎に C_Login～C_Logout を繰り返す。)

↓

C_CloseSession もしくは、 C_CloseAllSessions	⑦セッションの切断 C_CloseSession の場合 hSession：C_OpenSession で取得したハンドル C_CloseAllSessions の場合 slotID：C_OpenSession に指定したスロット ID

↓

終了処理	(2)終了処理参照

Ａ－３　CAPI 利用のシーケンス

(A) 初期化

CryptoAcquireContext	鍵コンテナのハンドルを生成する phProv,：関数アドレスリストポインタ pszContainer：NULL を指定する。 pszProvider：コンテナとして”HPKI Crypto Service Provider”を指定する dwProvType:PROV_RSA_FULL あるいは PROV_RSA_SIG を指定 dwFlags：０を指定

(B) 終了処理

CryptReleaseContext	鍵コンテナのハンドルを開放する phProv,：CryptAcquireContext で取得したハンドル dwFlags：０を指定

(C) 証明書取得

初期化処理	(A)初期処理参照

↓

CryptGetProvParam	CSP のパラメータの値を取得する。 hProv,：CryptAcquireContext で取得したハンドル dwParam：PP_CDERT_CHAIN（証明書） pbData：NULL pdwDataLen：長さを受け取るためのポインタを指定 dwFlags：PP_ENUMCONTAINERS｜PP_CRYPT_FIRST
CryptGetProvParam	CSP のパラメータの値を取得する。 hProv,：CryptAcquireContext で取得したハンドル dwParam：PP_CDERTCHAIN（証明書） pbData：データを受け取るバッファを設定 pdwＤａｔａＬｅｎ：受け取った長さを格納したポインタを指定 dwFlags：０

Ａ－１－２のシーケンスに従い、IC カード内のアプリケーションから全ての証明書を読み出す。ユーザの証明書以外の証明書を戻り値として返す。

↓

CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する hProv, : CryptAcquireContext で取得したハンドル dwKeySpec : AT_SIGNATURE（署名鍵）を指定

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。（IC カードに格納された利用者証明書(DER 形式)を取得 hKey：値を取得する鍵のハンドル pbData：NULL を設定する dwParam：KP_CERTIFICATE（利用者証明書） dwFlags： 0 を設定

ユーザの証明書を取得し、証明書の大きさを返す

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。（IC カードに格納された利用者証明書(DER 形式)を取得 hKey：値を取得する鍵のハンドル pbData：証明書を取得する領域のポインタ dwParam：KP_CERTIFICATE（利用者証明書） dwFlags： 0 を設定

ユーザの証明書を返す。

↓

CryptDestroyKey	鍵の破棄を行う。 hKey：鍵のハンドル

↓

ループ	(カウンタを初期値１に設定)

↓

証明書取得有無のチェック	求める証明書が取得できているかを判定する。 取得できていれば、終了処理へ。 取得できていなければ、次に進む。

↓

CryptGetProvParam	CSP のパラメータの値を取得する。(次のコンテナ) hProv, : CryptAcquireContext で取得したハンドル dwParam : PP_CERTCHAIN（証明書） pbData : NULL pdwDataLen : 長さを受け取るためのポインタを指定 dwFlags : PP_ENUMCONTAINERS
CryptGetProvParam	CSP のパラメータの値を取得する。 hProv, : CryptAcquireContext で取得したハンドル dwParam : PP_CERTCHAIN（証明書）

	pbData：データを受け取るバッファを設定 pdwＤａｔａＬｅｎ：受け取った長さを格納したポインタを指定 dwFlags： 0

エラーの場合にはループ終了（次のコンテナはない）

Ａ－１－２のシーケンスに従い、IC カード内のアプリケーションから全ての証明書を読み出す。ユーザの証明書以外の証明書を戻り値として返す。（返すための dwParam の値は、検討する必要がある）

↓

CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する hProv,：CryptAcquireContext で取得したハンドル dwKeySpec：AT_SIGNATURE（署名鍵）を指定

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。（IC カードに格納された利用者証明書(DER 形式)を取得 hKey：値を取得する鍵のハンドル pbData：NULL を設定する dwParam：KP_CERTIFICATE（利用者証明書） dwFlags： 0 を設定

ユーザの証明書を取得し、証明書の大きさを返す

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。（IC カードに格納された利用者証明書(DER 形式)を取得 hKey：値を取得する鍵のハンドル pbData：証明書を取得する領域のポインタ dwParam：KP_CERTIFICATE（利用者証明書） dwFlags： 0 を設定

ユーザの証明書を返す。

↓

CryptDestroyKey	鍵の破棄を行う。 hKey：鍵のハンドル

↓

ループの終わり	カウンタ i を 1 加算してループに戻る。

↓

終了処理	(B)終了処理参照

(D) 署名

パディングしたハッシュ値は、内部で計算するのではなく、HPKI アプリケーション側から

渡すものとする。

初期化処理	(A)初期処理参照

↓

CryptGetProvParam	CSP のパラメータの値を取得する。 hProv, : CryptAcquireContext で取得したハンドル dwParam : 証明書取得（CA） dwFlags : PP_ENUMCONTAINERS｜PP_CRYPT_FIRST

Ａ－１－２のシーケンスに従い、IC カード内のアプリケーションから全ての証明書を読み出す。ユーザの証明書以外の証明書を戻り値として返す。)

↓

CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する hProv, : CryptAcquireContext で取得したハンドル dwKeySpec : AT_SIGNATURE（署名鍵）を指定

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。 (IC カードに格納された利用者証明書(DER 形式)を取得 hKey：値を取得する鍵のハンドル pbData：NULL を設定する dwParam：KP_CERTIFICATE（利用者証明書） dwFlags： 0 を設定

ユーザの証明書を取得し、証明書の大きさを返す

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。 (IC カードに格納された利用者証明書(DER 形式)を取得 hKey：値を取得する鍵のハンドル pbData：証明書を取得する領域のポインタ dwParam：KP_CERTIFICATE（利用者証明書） dwFlags： 0 を設定

ユーザの証明書を返す。

↓

CryptDestroyKey	鍵の破棄を行う。 hKey：鍵のハンドル

↓

ループ	(カウンタを初期値１に設定)

↓

証明書取得有無のチェック	求める証明書が取得できているかを判定する。 取得できていれば、終了処理へ。

	取得できていなければ、次に進む。

↓

CryptGetProvParam	CSP のパラメータの値を取得する。（次のコンテナ） hProv, : CryptAcquireContext で取得したハンドル dwParam : 証明書取得（CA） dwFlags : PP_ENUMCONTAINERS

エラーの場合にはループ終了（次のコンテナはない）

Ａ－１－２のシーケンスに従い、IC カード内のアプリケーションから全ての証明書を読み出す。ユーザの証明書以外の証明書を戻り値として返す。

↓

CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する hProv, : CryptAcquireContext で取得したハンドル dwKeySpec : AT_SIGNATURE（署名鍵）を指定

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。（IC カードに格納された利用者証明書(DER 形式)を取得 hKey : 値を取得する鍵のハンドル pbData : NULL を設定する dwParam : KP_CERTIFICATE（利用者証明書） dwFlags : 0 を設定

ユーザの証明書を取得し、証明書の大きさを返す

↓

CryptGetKeyParam	鍵のパラメータの値を取得する。（IC カードに格納された利用者証明書(DER 形式)を取得 hKey : 値を取得する鍵のハンドル pbData : 証明書を取得する領域のポインタ dwParam : KP_CERTIFICATE（利用者証明書） dwFlags : 0 を設定

ユーザの証明書を返す。

↓

CryptDestroyKey	鍵の破棄を行う。 hKey : 鍵のハンドル

↓

ループの終わり	カウンタ i を 1 加算してループに戻る。

署名処理

CryptCreateHash	ハッシュオブジェクトの生成 Algid : アルゴリズム ID（将来は SHA2 ファミリを指定）

	hKey： 0 を設定 dwFlags： 0 を設定 phHash：ハンドル格納領域のポインタ

↓

CryptSetHashParam	ハッシュオブジェクトのパラメータを設定 hHash：ハッシュオブジェクトのハンドル dwParam：HP_HASHVAL を設定 pdData：ハッシュ値を設定

↓

CryptSignHash	ハッシュ値に署名を行う（） hHash：ハッシュオブジェクトのハンドル kwKeySpec：AT_SIGNATURE を指定 pbSignature：NULL を設定 pdwSigLen：署名データ長格納アドレス

↓

CryptSetProvParam	CSP のパラメータの値を設定する。 hProv,：CryptAcquireContext で取得したハンドル dwParam：PP_SIGNATURE_PIN pbData：PIN（NULL で終わる ASCII 文字列）へのポインタを指定 dwFlags： 0

CSP の U/I を利用する場合には、ここで PIN を設定しなくともよい。

↓

CryptSignHash	ハッシュ値に署名を行う hHash：ハッシュオブジェクトのハンドル kwKeySpec：AT_SIGNATURE を指定 pbSignature：署名データ格納アドレス pdwSigLen：署名データ長格納アドレス

ここで、Ａ－１－３のハッシュの設定、署名計算の演算を実行する必要がある。

↓

CryptDestroyHash	ハッシュオブジェクト破棄 hHash：ハッシュオブジェクトのハンドル

↓

終了処理	(B)終了処理参照

附属書 B（参考）　PKI アプリケーションの構造例

Ｂ－１．概要

複数の事業者が HPKI 認証局の運用を行うことを前提に、カードに搭載される PKI カードアプリケーションに関して、以下の条件の下で、相互運用性確保の仕様を検討する

- 各 HPKI 認証局が独自のカードを発行する。
- 証明書は、医療従事者等のエンドエンティティの証明書と認証局の証明書となる。
- HPKI で署名用証明書を利用した署名演算には、署名を行うたびに PIN の照合が必要とする。
- ファイル構造、コマンド、シーケンスは、標準に従う。

Ｂ－２．ファイル構造

ＤＦ／アプリケーションの構造は以下の通り

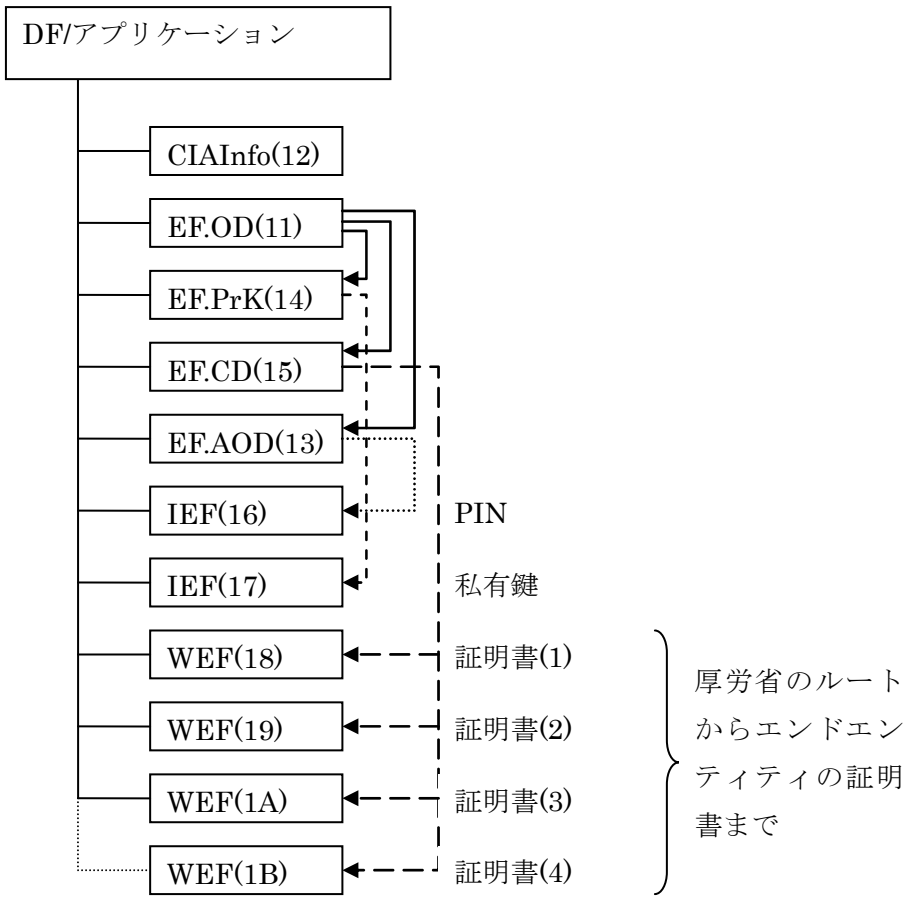

図 B-1　JIS X 6320-15 に準拠したファイル構造例

注 1）追加の信頼点の証明書に関しては、考慮していない。

注 2）カッコ内の数字(XX)は、短縮 EFID を示す。

注３）カード内に複数のアプリケーションが存在する場合でも、EF.DIR は存在しないものとする

Ｂ－３．PKI アプリケーションの識別（AID）

ISO/IEC 7816-15(JIS X 6320-15)に従う。RID は、ISO/IEC 7816-15 で決められるので、個別アプリケーション識別子(PIX)部分を各 HPKI 認証局が定める ID とする。

E8 28 BD 08 0F XX XX XX XX XX XX XX XX XX XX XX

7816-15 の RID　認証局から与えられる(最大 16 バイト)

注）XX の値は、厚生労働省が指定する。全体で最大１６バイトとなる。

Ｂ－４．各 EF の内容

ISO/IEC 7816-15 に従って規定する。すべて、短縮 EFID を用いる。

表 B-1　必要となるファイル一覧

ファイル	短縮 EFID	ファイルタイプ	内容
CIAInfo	‘12’ (ISO/IEC 7816-15(JIS X 6320-15)で指定)	透過	B-4.1
OD	‘11’ (ISO/IEC 7816-15(JIS X 6320-15)で指定)	透過	B-4.2：オブジェクト情報
AOD	‘13’	透過	B-4.3：認証オブジェクト情報
PrKD	‘14’	透過	B-4.4：私有鍵情報
CD	‘15’	透過	B-4.5：証明書情報
IEF	‘16’	Verify 用	PIN 格納
IEF	‘17’	RSA 用	私有鍵格納
WEF	‘18’	透過	証明書格納(エンティティ)
WEF	‘19’	透過	信頼点の証明書格納(HPKI ROOT 認証局の自己署名証明書)
WEF	‘1A’	透過	下位認証局への証明書格納
WEF	‘1B’	透過	下位認証局への証明書：追加

Ｂ－４．１　EF.CIAInfo

CIAInfo ::= {

version v2,

```
        label  “HPKI Application”, -- オプション
        cardflags { authRequired, prnGeneration } -- 暗号演算に利用者認証必要、乱数生成
                あり
}
```

Ｂ－４．２　EF.OD

```
authObjects : -- EF.AOD へのパス
        path : {
                efidOrPath '98'H -- EF.AOD の短縮 EFID ’13’
        },
privateKeys : -- EF.PrKD へのパス
    path : {
                efidOrPath 'A0'H -- EF.PrKD の短縮 EFID ’14’
    },
certificates : -- EF.CD へのパス
        path : {
                efidOrPath 'A8'H -- EF.CD の短縮 EFID ’15’
        }
```

Ｂ－４．３　EF. AOD

```
pwd : {   -- パスワード
        commonObjectAttributes {
                label "PIN ", -- オプション
                flags { modifiable }, --   オプション 変更可能
        },
        classAttributes {
                authId '16'H – 識別するための ID
        },
        typeAttributes {
                pwdFlags {
                        case-sensitive, --
                        local, -- このアプリケーション/DF 内でのみ有効
                        initialized -- 初期化されている
                },
                pwdType utf8, -- UTF8   文字セットは CryptAPI の条件により、ASCII となる
                minLength 4, -- 最小の長さ
                storedLength 16, -- 格納の長さ
                maxLength 16, --   最大の長さ
```

```
            pwdReference '96'H-- Verify コマンドの P2 パラメータ(アプリ固有の PIN)
                        IEF の短縮 EFID '16'
        }
}
```

Ｂ－４．４　EF.PrKD

```
privateRSAKey : {
        commonObjectAttributes {
            label "Private key of HPKI",
            flags { private },  -- 利用に認証が必要
            authId '16'H – 必要となる AOD エントリ
            userCosent 1 – 私有鍵によって署名を付加する際に、毎回認証を行う（追加）
              accessControlRules {
                    {
                        accessMode { execute }, -- 実行
                        securityCondition and : {
                            authId : '16'H, -- pointer to the AOD-entry
                            authReference : {
                                authMethod { userAuthentication }
                            }
                        }
                    }
                }
          }
          classAttributes {
                iD '17'H, -- 識別のための ID（EF-ID を利用し、証明書と共通にする）
                usage { nonRepudiation },
                },
          typeAttributes {
                value {
                    efidOrPath 'B8'H -- IEF の短縮 EFID '17'
                },
                modulusLength 1024
          }
}
```

Ｂ－４．５　EF.CD

```
x509Certificate : {
```

```
        commonObjectAttributes {
                label "HPKI END ENTITY CERTIFICATE",
        }
        classAttributes {
                iD '17'H --識別のための ID（私有鍵と共通にする）
        },
        typeAttributes {
                value indirect：
                path：{
                        efidOrPath 'C0'H --  格納された WEF の短縮 EFID '18'
                }
        }
},
x509Certificate：{
        commonObjectAttributes {
                label " MHLW CA CERTIFICATE ",   --厚労省の CA 証明書
        }
        classAttributes {
                iD '19'H --識別のための ID（EFID と同じとする）
                authority TRUE -- CA の証明書
        },
        typeAttributes {
                value indirect：
                path：{
                        efidOrPath 'C8'H --  格納された WEF の短縮 EFID '19'
                }
        }
}
x509Certificate：{
        commonObjectAttributes {
                label   " HPKI ROOT CA CERTIFICATE",   --各事業者の CA

        }
        classAttributes {
                iD '1A'H --識別のための ID（EFID と同じとする）
                authority TRUE – 厚労省の証明書
        },
        typeAttributes {
```

```
            value indirect：
            path：{
                efidOrPath 'D0'H – 格納された WEF の短縮 EFID '1A'
            }
        }
}
```

中間 CA が存在する場合にはもう 1 つ X509 オブジェクトを追加する

```
x509Certificate：{
        commonObjectAttributes {
            label " HPKI CA CERTIFICATE ",   --各事業者の中間 CA
        }
        classAttributes {
            iD '1B'H --識別のための ID（EFID と同じとする）
            authority TRUE – 中間 CA の証明書
        },
        typeAttributes {
            value indirect：
            path：{
                efidOrPath 'D8'H – 格納された WEF の短縮 EFID '1B'
            }
        }
}
```

附属書 C（参考）　PKI アプリケーション利用のコマンド

C－1　コマンド一覧

本ガイドラインでは、以下のコマンドについて満たすべき最低限の仕様を提示する。

SELECT FILE (ISO/IEC 7816-4)
VERIFY (ISO/IEC 7816-4)
READ BINARY (ISO/IEC 7816-4)
MANAGE SECURITY ENVIRONMENT (ISO/IEC 7816-4)
PERFOM SECURITY OPERATION (ISO/IEC 7816-8)

C-2 で各コマンドの最低限満たすべき仕様を示す。ステータスワードは、代表的な例であって、カードの実装に応じて JIS X 6320-4 で規定される他のステータスワードを適切に出力してもよい。

C－2　SELECT

C－2．1　条件

・ アプリケーション/DF のパーシャルセレクトが可能
・ FCI にて選択されたアプリケーション/DF の AID が取得可能

C－2．2　コマンドメッセージ

DF 選択

CLA	INS	P1	P2	Lc	データ	Le
00	A4	XX	XX			
(1)	(1)	(1)	(1)	(1)	(1～16)	(0 or 1)

EF の選択

CLA	INS	P1	P2	Lc	データ
00	A4	XX	XX		(EF-ID)
(1)	(1)	(1)	(1)	(1)	(2)

パラメータ	長さ	意味	備考
P1	1	選択制御子	
P2	1	選択オプション	
Lc	1	ファイル識別子 or ファイル名の長さ	
データ	1～16	ファイル識別子 or ファイル名	
Le	1	FCI データのバイト数	

P1 のコーディング

b8	b7	b6	b5	b4	b3	b2	b1	意味
0	0	0	0	0	0	x	x	ファイル ID による選択
-	-	-	-	-	-	1	0	カレント DF 直下の EF
0	0	0	0	0	1	0	0	DF 名選択

P2 のコーディング

b8	b7	b6	b5	b4	b3	b2	b1	意味
0	0	0	0	0	0	-	-	FCI オプションテンプレート応答
0	0	0	0	1	1-	-	-	FCI 応答なし(EFID による選択時)
0	0	0	0	-	-	0	0	最初または唯一のファイル
0	0	0	0	-	-	1	0	次ファイル（パーシャル DF 名指定可）

C－2．3 レスポンスメッセージ

レスポンス APDU

データ	SW1	SW2
(1)	(1)	(1)

パラメータ	長さ	意味	備考
データ	5～30	FCI(TAG='84')テンプレート	TAG='84'：DF 名
SW1	1		
SW2	1		

注）レスポンスの SW1 及び SW2 が'90 00'以外の場合のデータ長は、0 となる

データ部

6F	長さ	84	長さ	DF 名
(1)	(1)	(1)	(1)	(1～16)

C－2．4 ステータスワード

SW1	SW2	ステータスコードの意味
"90"	"00"	正常終了
"62"	"83"	選択された DF が閉そくしている 選択された EF の親 DF が閉そくしている
"67"	"00"	Lc および Le フィールドが間違っている。 APDU の長さが間違っている。
"68"	"81"	指定された論理チャネル番号によるアクセス機能を提供しない

	“82”	CLA バイトで指定されたセキュアメッセージング機能を提供しない
“６A”	“81”	機能が提供されない
	“82”	アクセス対象意ファイルがない
	“86”	P1 P2 の値が正しくない
	“87”	Lc の値が P1 P2 と矛盾している

C－３　VERIFY

C－３．１　条件

・ 短縮 EFID での実行

C－３．２　コマンドメッセージ

CLA	INS	P1	P2	Lc	データ
00	20	XX	XX		
(1)	(1)	(1)	(1)	(1 or なし)	(可変 or なし)

パラメータ	長さ	意味	備考
P1	1	なし（’00’固定）	
P2	1	参照データの限定	
Lc	1 or なし		
データ	可変 or なし	照合データ	

P2 のコーディング

B8	b7	b6	b5	b4	b3	b2	b1	意味
1	-	-	-	-	-	-	-	特定の参照データ
-	0	0	0	0	0	0	0	カレント EF
1	0	0	x	x	x	x	x	短縮 EF-ID（"11111”以外）

C－３．３　レスポンスメッセージ

レスポンス APDU

SW1	SW2
(1)	(1)

パラメータ	長さ	意味	備考
SW1	1		

SW2	1		

C－3．4　ステータスワード

SW1	SW2	ステータスコードの意味
“90”	“00”	正常終了
“63”	“00” “CX”	照合不一致とする 照合不一致[X によって、残りの再試行可能回数(0-15)を示す]
“67”	“00”	Lc および Le フィールドが間違っている。 APDU の長さが間違っている。
“68”	“81”	指定された論理チャネル番号によるアクセス機能を提供しない
	“82”	CLA バイトで指定されたセキュアメッセージング機能を提供しない
“69”	“81”	ファイル構造と矛盾したコマンド
	“83”	認証方法を受け付けない
	“84”	参照された IEF が閉そくしている
“6A”	“81”	機能が提供されない
	“82”	短縮 EF 識別子で指定した IEF がない
	“86”	P1 P2 の値が正しくない
	“87”	Lc の値が P1 P2 と矛盾している
	“88”	参照された鍵が正しく設定されていない

C－4　READ BINARY

C－4．1　条件

・ 短縮 EFID での実行

C－4．2　コマンドメッセージ

CLA	INS	P1	P2	Lc
00	B0	XX	XX	
(1)	(1)	(1)	(1)	(1)

パラメータ	長さ	意味	備考
P1-P2	各 1	読み出し対象短縮 EF-ID 及び先頭のバイナリデータのオフセット	

Le	1		

P1-P2 のコーディング

P1								P2	意味
B8	b7	b6	b5	b4	b3	b2	b1		
0	-	-	-	-	-	-	-	--	カレント EF 指定
0	x	x	x	x	x	x	x	xx	オフセット(15 ビット)
-	0	0	0	0	0	0	0	--	カレント EF 指定
1	0	0	x	x	x	x	x	--	短縮 EF-ID（'11111'以外）
1	0	0						xx	オフセット(8 ビット)

C－4．3　レスポンスメッセージ

レスポンス APDU

データ	SW1	SW2
(可変)	(1)	(1)

パラメータ	長さ	意味	備考
データ	可変	読み出されたデータ	
SW1	1		
SW2	1		

C－4．4　ステータスワード

SW1	SW2	ステータスコードの意味
“90”	“00”	正常終了
“62”	“83”	DF が閉そくしている
“67”	“00”	Lc および Le フィールドが間違っている。 APDU の長さが間違っている。
“68”	“81”	指定された論理チャネル番号によるアクセス機能を提供しない
	“82”	CLA バイトで指定されたセキュアメッセージング機能を提供しない
“69”	“81”	ファイル構造と矛盾したコマンド
	“83”	認証方法を受け付けない
“6A”	“81”	機能が提供されない
	“82”	短縮 EF 識別子で指定したファイルがない
	“86”	P1 P2 の値が正しくない
	“87”	Lc の値が P1 P2 と矛盾している

C－5　MANAGE SECURITY ENVIRONMENT

C－5．1　条件

・ 署名鍵の指定

C－5．2　コマンドメッセージ

CLA	INS	P1	P2	Lc	データ	Le
00	22	XX	XX			
(1)	(1)	(1)	(1)	(1)	(可変)	(1 or 3)

パラメータ	長さ	意味	備考
P1	1	‘41’: SET	
P2	1	SEID　あるいは　SET の場合は、テンプレートのタグ	‘B6’（電子署名用 CRT）
Lc	1	データ長	
データ	0 or 可変	CRT(可変の場合)	

C－5．3　レスポンスメッセージ

レスポンス APDU

SW1	SW2
(1)	(1)

パラメータ	長さ	意味	備考
SW1	1		
SW2	1		

C－5．4　ステータスワード

SW1	SW2	ステータスコードの意味
“90”	“00”	正常終了
“62”	“83”	DF が閉そくしている（カレント SE 利用できない）
“67”	“00”	Lc および Le フィールドが間違っている。 APDU の長さが間違っている。
“68”	“81”	指定された論理チャネル番号によるアクセス機能を提供しない
	“82”	CLA バイトで指定されたセキュアメッセージング機能を提供しない
“69”	“82”	セキュリティステータスが満足されない

	“85”	コマンドの使用条件が満足されない
“6A”	“80”	データフィールドのタグが正しくない
	“85”	Lc の値が、TLV 構造に矛盾している
	“86”	P1 P2 の値が正しくない

C－6　PERFORM SECURITY OPERATION

C－6．1　条件

- 私有鍵での電子署名暗号演算の実行
- パディングはカード外で行う

C－6．2　コマンドメッセージ

CLA	INS	P1	P2	Lc	データ	Le
00	2A	XX	XX			
(1)	(1)	(1)	(1)	(1)	(可変)	(1 or 3)

パラメータ	長さ	意味	備考
P1	1	‘9E’（ディジタル署名）	ENCYPHER 処理
P2	1	‘9A’：データフィールドが署名	
Lc	1	データ長	
データ	可変	署名されるデータ	
Le	1		

C－6．3　レスポンスメッセージ

レスポンス APDU

データ	SW1	SW2
(可変)	(1)	(1)

パラメータ	長さ	意味	備考
データ	可変	電子署名データ	
SW1	1		
SW2	1		

C－6．4　ステータスワード

SW1	SW2	ステータスコードの意味
“90”	“00”	正常終了
“62”	“83”	DF が閉そくしている（カレント SE 利用できない）

“67”	“00”	Lc および Le フィールドが間違っている。 APDU の長さが間違っている。
“68”	“81”	指定された論理チャネル番号によるアクセス機能を提供しない
	“82”	CLA バイトで指定されたセキュアメッセージング機能を提供しない
“69”	“82”	セキュリティステータスが満足されない
	“85”	コマンドの使用条件が満足されない
“6A”	“80”	データフィールドのタグが正しくない
	“85”	Lc の値が、TLV 構造に矛盾している
	“86”	P1 P2 の値が正しくない

付属書 D（参考）　IC カードリーダライタとのインタフェース

IC カードリーダライタとのインタフェースとしてプラットフォームにより以下の二つが存在する。

１）PC/SC　（MS-Windows におけるインタフェース）
２）PCSC-Lite(LINUX など UNIX 系の OS におけるインタフェース)

IC カードリーダライタが、PC などに接続される物理的インタフェースには、RS-232C、PCMCIA、USB 等がある。PC/SC や PCSC-Lite は、こうした物理的インタフェースの違いや、異なるベンダーの IC カードリーダライタの論理的なインタフェースの仕様の違いも吸収し、PKCS #11 などの IC カードに対応したモジュールに対して IC カードリーダライタへの汎用的な API を提供する。HPKI 用 IC カードガイドラインにおいても、特定の IC カードリーダライタへの依存性を最小限にするためにするため、PC/SC や PCSC-Lite といったフレームワーク上で動作することを推奨している。ここでは、PC/SC、PCSC-Lite、更に、USB インタフェースにおいて､IC カードリーダライタドライバの標準になりつつある CCID について概説する。

◇PC/SC

PC/SC は、ベンダー各社が製造する IC カード、IC カードリーダライタを、Windows 環境上で相互利用できるようにするためのインタフェース仕様として、米マイクロソフト社を中心とした複数のベンダーからなる PC/SC ワークグループによって定められ、1997 年 12 月にバージョン 1.0 が公開されている。
PC/SC においては、IC カードリーダライタのベンダーが、PC/SC に準拠した、個々の IC カードリーダライタのドライバを提供する必要がある。
IC カードリーダライタのアプリケーション(HPKI 用 IC カードガイドラインにおいては、PKCS #11 などの IC カードドライバ、ないしミドルウェア）は、PC/SC の API を利用することにより、個々の IC カードリーダライタのドライバを直接ハンドリングせず、ドライバから独立した実装が可能になっている。
Windows 環境では、マイクロソフト社の WHQL(Windows Hardware Quality Labs)が、IC カードリーダライタのようなハードウェアとドライバの認定を行っている。IC カードリーダライタと、対応するドライバに関して PC/SC に準拠した形での WHQL の認定が行われており、比較的安定した製品が数多く販売されている。

◇PCSC-Lite

PCSC-Lite は、LINUX など UNIX 系の OS において、PC/SC と同様の目的のために仕様が作成され、オープンソースソフトウェアとして提供されている。PCSC-Lite は、仕様的には、PC/SC のサブセットであるが、LINUX など UNIX 系の OS での実装可能な仕様となっている。PC/SC 同様、あるベンダーIC カードリーダライタを利用するためには、その IC カードリーダライタのためのドライバを PCSC-Lite に組み入れることになる。ただし、PCSC-Liteにおいては､マイクロソフト社のような認定制度(WHQLによる認定)は存在せず、また、多くの場合、IC カードリーダ・ライタベンダーは、ドライバを提供していない。代わって、オープンソースコミュニティが、PCSC-Lite 準拠した、IC カードリーダライタのドライバを提供している。そのため、現時点では、PCSC-Lite で動作させることのできる IC カードリーダライタは、PC/SC ほど多くはない。利用に当たっては、動作の範囲に注意する必要がある。

◇CCID

IC カードリーダライタの物理的インタフェースには､USB が非常に多く利用されつつある。USB の物理的インタフェースの IC カードリーダライタのインタフェースの仕様として、USB CCID (Chip/Smart Card Interface Devices)がある。USB CCID の仕様は、USB working group により 2001 年 3 月に Rev1.0、2005 年 4 月に Rev1.1 が発行されている。PC/SC に準拠した Windows 2000 以降のプラットフォームの USB CCID のドライバは、マイクロソフトにより提供されている。また、PCSC-Lite 準拠した USB CCID のドライバは、オープンソースコミュニティにより提供されている。そのため、USB CCID に準拠した IC カードリーダライタの場合、IC カードリーダライタ固有のドライバの提供を行う必要がない。こうしたことから､新しい USB の IC カードリーダライタの製品は､USB CCID に準拠したものが多い。

付録 1：参考文献

ISO/TS 17090-1:2002 Health informaticsw – Public key infrastructure – Part 1:Framework and overview

ISO/TS 17090-2:2002 Health informaticsw – Public key infrastructure – Part 2: Certificate profile

ISO/TS 17090-3:2002 Health informaticsw – Public key infrastructure – Part 3: Policy management of certification authority

PC/SC Specifications　http://www.pcscworkgroup.com/specifications/overview.php

付録２：作成者名簿

作成者（五十音順）

浅野　之治	凸版印刷㈱
遠藤　方洋	凸版印刷㈱
川村　眞知子	富士通㈱
菅野　好史	㈱ＮＴＴデータ
喜多　紘一	東京工業大学
齋藤　和美	三菱電機㈱
酒井　正仁	大日本印刷㈱
半田　富己男	大日本印刷㈱
牧野　智謙	大日本印刷㈱
町田　悦郎	（財）医療情報システム開発センター
松本　泰	セコム㈱
宮崎　徹也	日本データカード㈱
茗原　秀幸	三菱電機㈱
谷内田　益義	㈱リコー
米田　健	三菱電機㈱

（JAHIS 標準 08-002）

2008 年 6 月 発行

～HPKI 対応 IC カードガイドライン～

発行元 保健医療福祉システム工業会

〒105-0001 東京都港区虎ノ門 1 丁目 19－9

（虎ノ門 TB ビル 6F）

電話　03-3506-8010　　FAX　03-3506-8070