National Institute of
Standards and Technology
Technology Administration
U.S. Department of Commerce

NISTIR 7284

Personal Identity Verification Card Management Report

Jim Dray

David Corcoran

NISTIR 7284

Personal Identity Verification Card Management Report

Jim Dray

David Corcoran

C O M P U T E R S E C U R I T Y

Information Technology Laboratory

National Institute of Standards and Technology

Gaithersburg, MD, 20899-8940

January 2006

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Michelle O'Neill, Under Secretary for Technology

National Institute of Standards and Technology
William A. Jeffrey, Director

REPORTS ON COMPUTER SYSTEMS TECHNOLOGY

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of non-national security-related information in federal information systems. This Interagency Report discusses ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report, 35 pages

(January 2006)

Acknowledgements

The authors, Jim Dray of the National Institute of Standards and Technology (NIST) and David Corcoran of Identity Alliance, wish to thank their colleagues who reviewed drafts of this document and contributed to its development. The authors also gratefully acknowledge and appreciate the many contributions from the public and private sectors for the continued interest and involvement in the development of this publication.

Abstract

NIST Special Publication 800-73 (http://piv.nist.gov) provides technical specifications for Personal Identity Verification (PIV) cards. However, it does not contain a complete card management specification for PIV systems. This report provides an overview of card management systems, identifies generic card management requirements, and considers some technical approaches to filling the existing gaps in PIV card management. The primary guiding principles in selecting technical approaches for consideration are that they should require no changes to the existing PIV specifications and should conform to applicable formal standards. Expanding the PIV command set to include management and personalization would result in a higher level of consistency and testability for PIV card issuance processes, enhanced ability to outsource various card management components and functions, and improved overall security for the federal PIV framework.

Table of Contents

List of Figures

List of Tables

1. Introduction

The Homeland Security Presidential Directive (HSPD-12) has been used as a catalyst for the development of standards governing the interoperable usage of identity credentials for physical and logical access purposes across all agencies of the Federal Government. The resulting suite of National Institute of Standards and Technology (NIST) specifications defines a collection of processes through which a federal employee or contractor is first identified and then issued a smart card based credential, which can be used in an interoperable manner across federal agencies. These smart cards are referred to as Personal Identity Verification (PIV) cards.

The scope of the PIV specifications was defined by a set of use cases developed among the federal agencies during the inception of NIST's response to HSPD-12. Use cases specifically address only credential usage and consider the personalization and management of smart card based credentials out of scope. This report examines card management requirements, presents these requirements in the context of PIV systems, and reviews technical approaches to meeting these requirements. The motivation for this work is to support and enhance interoperability of PIV cards. Key high-level concepts are presented in *italics.*

2. Overview of Card Management Elements

Card management is the term applied to the preparation of a smart card token before its issuance to the cardholder and the provision of administrative functions for the card during its lifetime of use in the possession of the cardholder. Card management is particularly important because it so basically affects the security characteristics of the card, and security is the *raison d'etre* of smart cards. In this section, the basic elements of card management will be considered.

2.1 Card Personalization/Management Lifecycle

The phases through which a card passes—from initial development and manufacturing, to issuance to the end cardholder, to final retirement of the card—are often referred to as the card's *lifecycle*. *Card management* is a general term used to describe the system through which a card is managed—that is, through which various pieces of information on the card are initialized, monitored, and/or updated while the card is in routine use. Personalization is the process through which the card is bound or personalized to its owner (i.e., cardholder) before the card being issued—that is, placed in the physical possession and control of the cardholder. Card management systems (CMS) usually provide mechanisms by which a smart card is personalized. A distinguishing example is the personal identification number (PIN) unblock feature that is available in many CMSs. PIN unblock typically consists of a process through which a blocked card is unblocked and then updated with a new PIN provided by the user.

Smart cards must be managed during the various stages of their lifecycle. This section describes the various stages of the card's lifecycle and what types of operations occur at these particular stages. Figure 2-1 illustrates the three stages of card management that occur during the lifecycle of a smart card; some systems define more stages but most include at least these three.

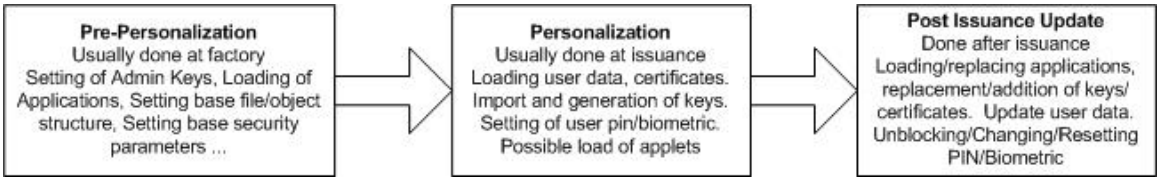

Figure 2-1: Smart Card Management Stages

- **Pre-personalization**. The first stage occurs at the factory. This stage is agreed on by the card producer and the customer and usually consists of setting up some core applications and generic data models to prepare the card for issuance by the customer through his or her CMS. Soft masks (firmware) may be applied during this phase, as well as loading card applications for multi-application cards. By preloading card applications, typically an issuer can trim several minutes off the issuance process. Administrative keys are set as agreed by the two parties through a previous key generation and exchange ceremony. The card may then be preloaded with a skeleton file/object structure and generic data. The end result of the pre-personalization process is to create a "common" card that is used as the starting point for the personalization process, in which every card is made slightly different from all other cards by including "personalization" information that ties a specific card to a specific cardholder.
- **Personalization.** This stage often occurs when the card is on customer premises and usually takes place right before or during the issuance process. Personalization will require more

cardholder-specific information than pre-personalization. Certificate/key generation and loading occurs. The user's PIN and/or biometrics are captured and loaded onto the card. Demographic data about the user may also be loaded.

- **Post-Issuance.** Post-issuance updates occur after the card has been issued and may involve updating any information on the card, including card applications, PINs, biometrics, demographic data, certificates, and keys. A good CMS will have a seamless way for updating the card once it has been issued. This is key to preventing the costly reissuance of cards as standards and data models change on the card.

The following operations are included in the management and personalization process:

- Loading and updating card data objects
- Key generation
- Key import
- Loading card applications
- Unblocking PINs
- Changing PINs.

The smart card is the ultimate protector of its data and communication paths. On many multi-application smart cards, this is equally true of the card application(s). Publishing a complete card command set specification should not be considered a security vulnerability because the card or card application will protect its data when implemented properly. The base concept is that security is grounded in proper implementation of well defined protection mechanisms, and knowledge of the mechanisms actually used should not provide would-be attackers with any significant advantage. Knowing how to perform a management or personalization command sequence with a card does not equate to having the privilege to do so.

When considering ubiquitous mass issuance of smart card based identity tokens, components must be amenable to procurement from multiple vendor sources, each of which meet strict interoperability guidelines so that components can be mixed and replaced over time. Although historically it was difficult to imagine a smart card system in which the CMS, card applications, and post-issuance management system were produced by different vendors, this is the goal of many current smart card deployments. Single vendor systems should not be required to achieve large-scale, working deployment. The identity token (card) might be used in environments far removed from where the card originated, so it must adequately protect itself and make certain capabilities available to different systems. This is an issue when, for example, a government employee roams across the security infrastructures of different agencies. *The ability to use PIV credentials across agency boundaries is a primary HSPD-12 requirement.*

A smart card's card interface specification (i.e., the command set it provides) and its data model must be known to the CMS, the card operating system or application, and the middleware component(s) that enable card usage with host systems. If the card is used for physical access purposes, this card interface may also need to be known to the physical access system. Specifically, for card management and personalization, the CMS and card operating system or application must share knowledge of the card interface specifications responsible for personalization and management. Without complete specifications, the CMS and card operating system (or application) are tightly bound, requiring implementation by a single vendor. If a CMS vendor wishes to support multiple card operating systems

or card applications, the vendor must have specific code for each in order to perform management and personalization functions. By fully specifying card management interfaces, this tight integration requirement can be avoided and the potential for a CMS to work with new cards and their applications would be primarily a matter of testing and validation.

An example in which a smart card needs to be personalized or managed outside its originating environment is a situation in which a cardholder must be given temporary credentials to access logical facilities off-site. In this example, the need to place data such as certificates and keys on the card temporarily might be a critical requirement. Personalization or management of a smart card is controlled by the issuing authorities' policies of whether the card can be updated outside its environment. It is possible to allow only certain cards to have this feature.

The CMS acts as a broker between all other parties in an identity management system and the card. It is responsible for establishing a chain of trust from initial card issuance to management of the card in the field. It may communicate with various PKI components such as the certificate authority and directory service. It may also communicate with one or more Physical Access Control Systems (PACS). Figure 2-2 illustrates how the CMS and its associated Hardware Security Module (HSM) act as an arbiter between outside systems and the smart card.

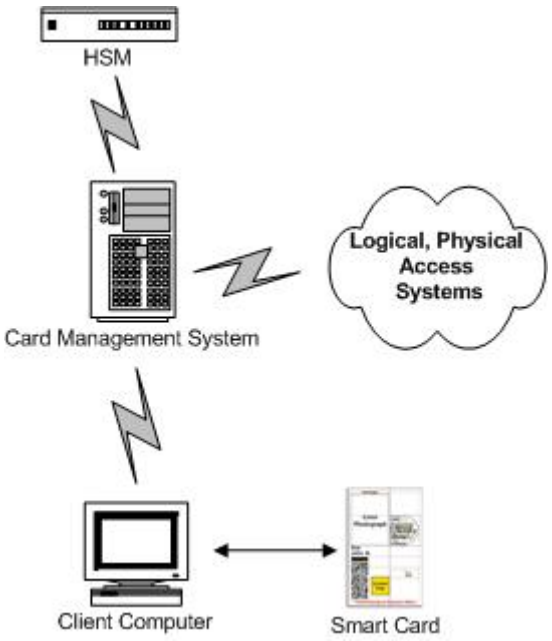

Figure 2-2: Card Management System Component Interactions

2.2 Card Management Operations

The CMS requires an ability to write and update data on the smart card during the various lifecycle phases. Certificates will be written to the card and updated if a user's public key infrastructure (PKI) credentials have to be replaced before card termination. Demographic data and other data also may need to be written and updated. Typically, write operations are protected by some mechanism such as verification of a PIN or external authentication using a set of keys and a challenge response protocol. Issuing authorities that wish for their cardholders to be able to update their cards without the use of a CMS might choose to allow writing to the card using only a PIN. Those issuers who do not wish for users to write to the card may require an administrative PIN or an external authentication. A card also

might support both methods so that some data objects might be updated via PIN, some by external authentication, and some never or always (without external authentication).

2.2.1 Key Generation: Logical Access Credentials for Authentication Purposes

The issuing authority authenticates to the card to enable key pair generation. The card is asked to generate a key pair using the existing PIV Generate Asymmetric Key Pair command. The generated public key is returned to the relying application or CMS, which wraps that public key into a request for certificate generation. The request is sent to a certificate authority, which can then approve it. The certificate authority generates a certificate containing the public key provided by the card, based on a known certificate profile. The certificate is returned to the application or CMS, which then updates the card through, for example, the PIV Put Data command. Optionally, the card's cryptographic information discovery mechanism is updated to note that a new key-pair and certificate are now available on the card.

2.2.2 Key Import: Import of Key Management Credentials

The CMS generates a public/private key pair using a secure HSM device. The private key typically is escrowed for possible future recovery in the event of a lost, stolen, or reissued card. The public key is wrapped into a request for certificate generation. This request is sent to the certificate authority, which can then approve the request. It generates a certificate based on the known certificate profile, as well as the public key provided by the HSM. The certificate is returned to the CMS, which imports the private keys into the card (this operation is not yet defined in Special Publication [SP] 800-73). It also writes the certificate to the card. Optionally, the card's cryptographic information discovery mechanism is updated to note that a new key pair and certificate are now available on the card.

2.2.3 Loading Card Applications

Loading and managing card applications represent crucial security points during the card's lifecycle. During these processes, the system could be compromised, which could affect the entire population of issued cards—in essence, rendering the entire system suspect and unusable. It is extremely important that the security of the protocols and systems that manage the on-card applications be sound and well evaluated. Many multi-application smart cards are only as secure as the card application that is running on them. If that card application were replaced, intentionally or inadvertently, an opportunity might occur for an on-card Trojan horse. There would be the potential for attackers to locate secret backdoors into which they could gain unauthorized access to card content once the cards were fielded.

In the past, card manufacturers relied on highly secretive and proprietary methods for managing card applications. More recently, standards have focused on card application management—for example, Global Platform (previously, Open Platform) [8]. Global Platform provides a standard mechanism for loading and managing card applications and their security policies. It also provides a mechanism to securely authenticate and exchange encrypted messages with the card so that remote management might be possible. Even more recently, many of the methods defined in Global Platform have become the basis for a new International Organization for Standardization (ISO) specification: ISO 7816-13 [3].

2.2.4 PIN Unblock Using a Remote Card Management System

The user PIN will become blocked after a certain amount of unsuccessful verification attempts. Once the PIN is in the blocked state, it can no longer be verified until the PIN is unblocked. This action usually occurs by communicating to a remote CMS. Several ways exist in which PIN unblock can occur. This section will discuss three major ways in which PIN unblocking can be implemented.

2.2.5 PIN Unblock With Unblocking PIN

An unblock PIN, or PIN Unlock Key (PUK) is an administrative PIN specifically used to unblock another PIN when it becomes blocked. The unblock PIN may be transmitted over a secure connection from the CMS to the card. Alternatively, the unblock PIN might be provided over web, e-mail, or even the telephone. An unblock PIN should be used over a secure channel to the card, especially if the unblock PIN is shared among multiple cards. Alternatively, each card might contain a unique unblock PIN that is changed each time an unblock operation occurs. Given that the unblock PIN is transmitted directly to the card, a secure communications channel may be required between the CMS and the card to prevent disclosure of the PIN.

2.2.6 PIN Unblock using External Authentication

Another method used for PIN unblocking is to execute a remote cryptographic challenge/response authentication between the CMS and the card. This method provides better security in that there is no static unblock PIN transmitted to the card; rather, a cryptographic challenge/response protocol is used that exposes no sensitive information.

PIN Reset is used when a successful unblock occurs. The user must provide a new PIN, which is then used to replace the old PIN that had been forgotten.

PIN Release is used when a successful unblock occurs and the old, forgotten PIN is provided back to the client. Depending on implementation, this may then be used with a Change PIN operation to change the PIN to a new value, or the old PIN might be provided and used for future operations.

2.2.7 PIN Change

Typically, PIN change operations can occur in the client middleware and have no interaction with card management or post-management systems. PINs are usually created and modified based on a policy that mandates the strength and composition of the PIN. PINs may need to contain a mixture of characters, and may need to be one not within the past several PINs used. Other policies might mandate the length and strict composition of the PIN (e.g., numeric) so it can be entered into an external smart card reader with PIN pad device.

3. Selected Potential Extensions for SP 800-73

The current version of the SP 800-73 specification does not fully address card management functions. Extensions to the PIV card interface specification would be needed to provide interoperability in this area. In this section, some of the necessary enhancements will be considered.

3.1 Cryptographic Object Discovery

The current SP 800-73 provides a fixed mechanism for the cryptographic data model that the PIV card application implements. Middleware implemented to this specification may have its own discovery mechanisms extending the current PIV scope, as well as testing for the existence of mandatory and nonmandatory cryptographic objects. For example, to determine whether the PIV Key Management certificate exists in a PIV application, the middleware must perform a Get Data command on that object's defined tag. This approach has considerable overhead because it is necessary to attempt to read each optional object on the card to determine its existence. A better method consists of having a cryptographic information structure on the card that can be quickly read to determine what cryptographic objects exist.

An efficient, standardized object discovery mechanism is essential to future expansion of the PIV framework because this will improve cross-agency interoperability, minimize costs arising from technology migration, and facilitate the PIV systems' ability to support additional application requirements.

NIST SP 800-78 defines a migration path for upgrading the PIV cryptographic algorithm suite in accordance with federal cryptographic policies and mandates. Standard cryptographic object discovery mechanisms will ease this migration. It also is highly likely that agencies will want to use the PIV card platform for purposes beyond its intended use for logical and physical access and that these purposes will require cryptographic discovery mechanisms.

The Government Smart Card Interoperability Specification (GSC-IS), National Institute of Standards and Technology Interagency Report [NISTIR] 6887, defines a discovery mechanism known as the Card Capabilities Container (CCC). [9] The CCC contains information regarding the types of applications on the card and a mechanism for translating card command sets to support various card interfaces. PIV cards have a CCC that can be used to identify data objects, but the PIV version of the CCC does not support the GSC-IS card command set translation mechanism.

SP 800-73 provides an explicit definition of the card interface and namespace for card data objects to meet the stringent interoperability requirements of HSPD-12. Interoperability can be further enhanced by adding a cryptographic discovery mechanism.

The problem of cryptographic object discovery is not new. Rivest, Shamir, and Adleman (RSA) security produced the Public-Key Cryptography Standards (PKCS) #15 industry standard, which some vendors began to use as a common, interoperable format for storing and discovering cryptographic objects on smart cards. [10] Organizations sometimes produced "lite" versions that were not always interoperable with other products claiming PKCS#15 conformance.

The idea behind PKCS#15 is simple: Place an object on the card in a known location that references other objects that provide information about what exists on the card and how one can access it. PKCS#15 encodes its information using the ASN.1 [http://www.asn1.org/] syntax so the information is small and portable. With heterogeneous computing environments, it is important that the endianness(byte and bit

order) of data be understood and well defined. Mixing endianness within data structures on the card should be avoided as much as possible. Network byte order or big endian format would be most common. With various new smart card technology such as 1 Megabyte cards becoming available, it is important that data be represented to allow various lengths and that design decisions not be constrained by current technology limitations. In addition, ASN.1 libraries are widely available; are stable; and are often free, open source software.

PKCS#15, which has been widely accepted as a cryptographic discovery mechanism, has been adopted in the international standards community as an ISO specification: ISO 7816-15 [3]. It has become the basis for interoperability for various national identity programs in Europe and Asia, including the following:

- Finnish ID Card
- Swedish ID Card
- Belgian ID Card
- Taiwanese ID Card.

The Finnish Electronic ID Card, FineID, is one of the most comprehensive specifications [5] based on PKCS#15. It describes in great detail not only each PKCS#15 object but also the Access Control Rules (ACR) for card management and use.

Note that PKCS#15 and ISO 7816-15 will be referenced as 7816-15 throughout the remainder of this document because the PKCS#15 standard has been transitioned into the ISO standards group.

3.1.1 ISO 7816-15 Structure

Figure 3-1 illustrates the ISO 7816-15 file/object hierarchy.

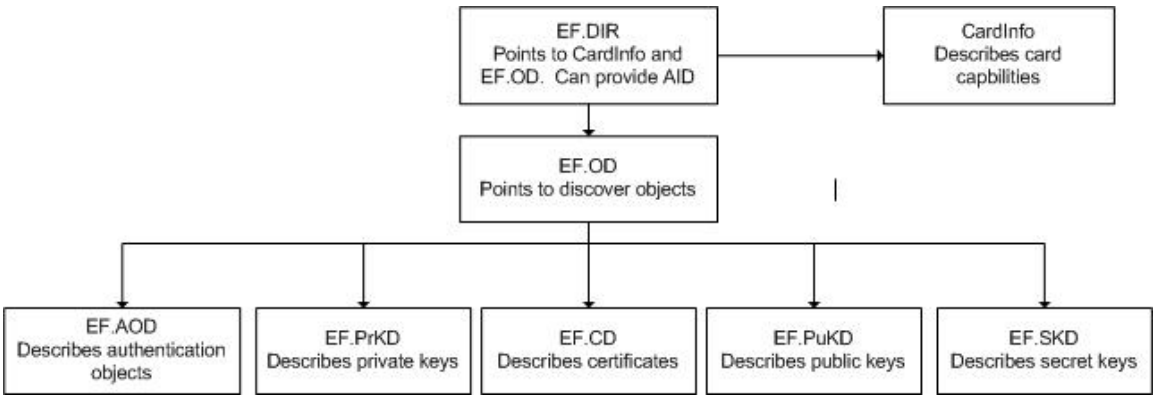

Figure 3-1: ISO 7816-15 File/Object Hierarchy

To use this information to locate a data object on the card, the middleware would first read the EF.DIR. This object contains information that allows the middleware to find the CardInfo object, the EF.OD, and potentially the Application Identifiers (AID) of the application that contains the data object. The CardInfo object contains information specifically about the card (e.g., version number, card characteristics, serial number, manufacturer/issuer identification, card label, supported algorithms, time of last update and potentially other information), with the minimum mandatory elements being the version

number and card characteristics. The EF.DIR also points to the EF.OD, which is an object that describes where the other discovery objects are stored on the card. Some of the discovery data structures that the EF.OD can point to are as follows:

- EF.AOD—Authentication Objects (e.g., PINs, biometrics, card authentication keys)
- EF.PrKD—Contains a list of private keys
- EF.CD—Contains a list of certificates
- EF.PuKD—Contains a list of public keys
- EF.SKD—Contains a list of secret keys
- Other cryptographic information objects that can be used to extend the discovery capabilities.

Each discovery object contains a listing of other objects available on the card. Take the example of a PIN used to authenticate the cardholder to perform certain restricted operations. This PIN can be discovered by reading the EF.AOD (Authentication Objects). The AOD would contain an entry describing minimum and maximum lengths for the PIN, the character set that must be used, flags to note if it needed padding, and a reference number that could be used when calling certain functions (e.g., Verify or Change Reference Data).

Suppose the example of key recovery is used as noted earlier in this document. In this scenario, one is introducing a new key and certificate to the card. Once the key and certificate are added to the card, the middleware would then update the EF.PrKD to add the private key to be discovered and then optionally add the certificate and public key to the EF.CD and EF.PuKD, respectively. Middleware that properly reads these discovery structures would then be able to learn that that this new cryptographic material exists and would be able to provide it to be used by upper level applications.

7816-15 provides mechanisms for a wide array of cryptographic material to be discovered. This could add complexity if the entire 7816-15 structure were required. To enhance interoperability, it is useful to specify a minimum mandatory subset of 7816-15. Through the use of ASN.1, other nonmandatory features of 7816-15 could be used without breaking interoperability. Section 6 defines an example subset of 7816-15 that might be considered for minimum interoperable PIV usage.

Given the flexibility of ASN.1, it is plausible for parties to extend the minimum, mandatory subset with additional capabilities available in 7816-15. For example, it may be desirable to have the smart card store a listing of trusted root certificate authorities for doing local path validation. 7816-15 includes extensions for discovering trusted certificates, and supporting this should not affect other mandatory structures.

3.1.2 Simplified ISO 7816-15 Structure Mapped into Tags for Get Data

To adapt 7816-15 to the existing namespace with SP 800-73, a new tag must be defined that can be used to read and write the EF.DIR. Given that the EF.DIR indicates how to access other objects in the 7816-15 structures, the other 7816-15 object tags would not necessarily need to be implicitly defined as reading the EF.DIR could discover them. Table 3-1 provides one possible tag set for 7816-16 objects on PIV cards. Note that only the EF.DIR would be mandatory.

Table 3-1. ISO 7816-16 Objects on PIV Cards

7816-15 Structure Object	Tag Value (for Get/Put Data)
EF.DIR	5FC201
EF.CARDINFO	5FC202
EF.OD	5FC203
EF.AOD	5FC204
EF.PRKD	5FC205
EF.CD	5FC206
EF.PUKD	5FC207
EF.SKD	5FC208

The 7816-15 objects could be accessed using pivGetData and pivPutData through the SP 800-73 Part 3 API. This would require Object Identifiers (OID) that could be passed to these respective functions. Table 3-2 shows an example of how this could be achieved:

Table 3-2. OIDs for ISO 7816-16 Objects

7816-15 Structure Object	OID Value (for pivGet/PutData)
EF.DIR	2.16.840.1.101.3.7.2.2.1
EF.CARDINFO	2.16.840.1.101.3.7.2.2.2
EF.OD	2.16.840.1.101.3.7.2.2.3
EF.AOD	2.16.840.1.101.3.7.2.2.4
EF.PRKD	2.16.840.1.101.3.7.2.2.5
EF.CD	2.16.840.1.101.3.7.2.2.6
EF.PUKD	2.16.840.1.101.3.7.2.2.7
EF.SKD	2.16.840.1.101.3.7.2.2.8

3.1.3 7816-15, OIDs, and Tags:

The SP 800-73 namespace uses tags to label data objects. 7816-15 uses the notion of a Path, which is simply an octet string. A Path in 7816-15 is intended to be used with a file system type structure, although given the generic definition of a Path as an octet string one could encode other values within that field. For example, the SP 800-73 data object tags could be represented in the Path even though 7816-15 does not specifically address naming data by tags or OIDs.

An example 7816-15 EF.CD PIV certificate in ASN.1 syntax is show below:

30 66

[

30 33

Label: “PIV Authentication Certificate”

0C 1E

50 49 56 20 41 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 20

43 65 72 74 69 66 69 63 61 74 65

ACR: “Read Always, PIN to Update”

30 11

30 06

03 02 00 80 05 00

30 07

03 02 00 40 04 01 80

Identifier: 36 Byte UID

30 26

04 24

29 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB B3 A6 DB

3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE B3 12 4D C8

Referenced Value “5FC105”

A1 07

30 05

04 03

5F C1 05

]

3.1.4 Monolithic 7816-15 Structures

To reduce the number of data objects required for discovery and still use the features of 7816-15, it is possible to represent the contents of the discovery data structures within the EF.OD itself. There are some drawbacks to this approach because it requires a single data object to hold the discovery data for all objects, which can become burdensome for the parser if it is only interested in certain data types. This method also requires that the access control rules of the EF.OD be identical to those that all of the discovery data structures would have. For example, there could be no granularity in access control for each of the discovery data structures, as they would be contained in a single data object.

3.1.5 Transitioning from the CCC to 7816-15

Transitioning from the CCC to 7816-15 should be easy given that the two discovery mechanisms can co-exist. The CCC even includes a field that denotes whether a 7816-15 structure exists on a card. The middleware discovery process could take the following steps:

- Attempt to read the CCC and its contents.
- If the first step fails, read the standard EF.DIR location and its contents.
- If the first step succeeds, determine if the PKCS#15 flag is set in the CCC.
- If there is no PKCS#15 flag, use the CCC for discovery.
- If the PKCS#15 flag is set, read the standard EF.DIR location and its contents.
- If neither the CCC nor 7816-15 structures exist, rely on the fixed, standard locations of the default PIV objects only.

3.2 Secure Messages/Channels

Security should be implemented end to end between communicating parties to lessen the threat of security protocol attacks. During an attack, the adversary will look to exploit the weakest link in the security protocol. Making assumptions about the security of a host platform can be dangerous and should be avoided, especially when the smart card has the capability to provide end-to-end security mechanisms.

One such end-to-end security mechanism is the use of secure messaging or secure channels with the smart card as one endpoint of the secure communications path. This allows the CMS to establish a secure pipe from the CMS directly to the card instead of just to the host connected to the card. Secure messaging not only protects the confidentiality of the command-response but also establishes data authentication so that each party can trust the source of the data.

Smart cards provide secure messaging mechanisms that protect the confidentiality of the data from the CMS, authenticate both parties, and can include message digests to ensure the protected data has not been altered. These mechanisms should be used any time private data is communicated with the card such as during key injection procedures, application loading, PIN handling, and when exchanging private demographic data.

Post-issuance updates can reduce the cost of changes to issued cards, by updating the cards' application and/or data objects securely and transparently over a network. This significantly reduces, and occasionally eliminates, return visits to issuance stations and reissuance of cards. To deter fraud and PIV credential tampering through abuse of post-issuance update mechanisms, these mechanisms should be standardized using strong authentication and secure communications.

Two commonly used methods for secure communications with the card have been standardized. The first is Global Platform [8] secure channels. GP secure channels initially provided only confidentiality in the outgoing direction. Recent updates of the standard added support for bidirectional encryption of the commands being sent. Global Platform [8] is supported by nearly all CMSs deployed today.

ISO 7816-4 [3] provides a mechanism known as secure messaging, which is the ISO version of secure channel capability. With any form of secure messaging, the protocol must communicate the algorithm, mode of operation, key reference, and input data to be used. Using secure messaging and secure channels is therefore a reasonably complicated process. Given the migration of Global Platform [8] into ISO as

ISO 7816-13 [3] and its potential usage of secure messaging, it would be advisable to explore ISO secure messaging for potential future usage by PIV card applications and external management software. A profile defined over ISO 7816-13 should be the basis for PIV post-issuance update mechanisms, once 7816-13 has been formally approved by ISO.

3.3 Key Injection

SP 800-73 specifies a nonmandatory Key Management Key and certificate that can be used for encryption to support secure exchange of key material. Key Management keys hold the burden that if they are lost, the key material encrypted with those keys cannot usually be recovered unless the Key Management key was escrowed during issuance or if the key material was backed up by another means. A key that must be escrowed is usually generated off the card, typically through an HSM, and then securely injected into the card through some secure path. SP 800-73 does not describe how this key can be injected into the card in an interoperable manner. A very simple mechanism could be devised that defines new tags for these keys and denotes the format in which they are to be sent to the card, as follows:

Table 3-4. New Tags for Key Injection

Tag	Value	Comment
9D06	RSA Key Import BER-TLV Objects	PIV Key Management Key

RSA Private Key Components and Tag Values:

Table 3-5. Tag Values for RSA Key Components

Tag	Value
40	Public Exponent
41	Public Modulus
42	Private P
43	Private Q
44	Private DP1
45	Private DQ1
46	Private PQ

The following describes the P, Q, DP1, DQ1, and PQ parameters:

- P is the smallest RSA prime
- Q is the other RSA prime
- DP1 = D mod (P-1), where D is the secret RSA exponent
- DQ1 = D mod (Q-1)
- PQ = P^{-1} mod Q.

To inject the key onto the PIV card, the PutData card interface command would be invoked in the chaining mode and would send the noted tag and its subcomponents depending on whether the key was to be injected in CRT or modulus/exponent format. A similar model might also be adapted to define how ECC, DES, AES, and other keys would be injected onto the card.

Given that the key must be transmitted to the card from a remote location during key injection, it is crucial that the security of that key cannot be compromised. Secure messaging is a fundamental requirement for key injection.

3.4 Object Creation

3.4.1 PIV Mandatory Data Objects

PIV issuance systems are responsible for binding the identity of the cardholder to a set of credentials, and loading those credentials onto a PIV card. The format and content of the credential objects on a PIV card are clearly defined in the PIV suite of standards and specifications, and PIV issuance systems must generate credential objects that conform to those specifications.

It is not currently possible to test PIV issuance systems for conformance to interface specifications because those specifications have not yet been defined. At the time of this writing, PIV issuance systems are free to use any method to load PIV data objects onto PIV cards as long as that method provides the level of assurance required by FIPS 201 and other relevant federal security standards and policies.

However, it is possible to test the ability of PIV issuance systems to generate credential objects that conform to the PIV Data Model in a black box sense. This effort would be accomplished by verifying the format and content of credential objects generated by a given issuance system against the PIV Data Model specifications.

3.4.2 Extended Objects

The PIV System will become an integral part of the federal IT infrastructure. Additions and changes to the PIV Data Model are inevitable over the lifespan of the PIV program. To accomplish this, PIV issuers need a standardized way to extend the current Data Model by adding new objects. This requires a framework for generating new identifiers, defining access control rules, and specifying the format and content for these objects.

3.4.3 Namespace for Extended Objects

The term “namespace” refers to a scheme for labeling data objects. The management and control of namespaces can be one of the most important roles in an identity card deployment. Strict definition of how labels are created as well as who can use labels and what they can be used for is vital. In conception of a namespace management program, it is easiest to define only what is mandatory and restrict anything else.

Even with a strict namespace management scheme, there will always be the need and desire to extend the namespace to accommodate additional data objects. Beyond mandatory data objects, there can also be mandatory optional and purely optional objects. Mandatory optional refers to objects that must be uniformly named and encoded but whose existence is not mandatory. Optional objects may exist but have no defined interoperability format.

The current philosophy behind SP 800-73 is that any data objects outside the scope of FIPS 201 and SP 800-73 should be contained outside the PIV application. PIV takes the strict namespace management

approach. As the PIV application evolves, it might be possible to allocate a namespace for extended data objects to create a single on-card application that can be fully functional inside and outside of the PIV realm. To allow for extensions to the PIV namespace, there must be some basic management of this namespace so that as NIST and government agencies expand the scope of PIV, the risk of namespace conflicts is mitigated.

The current methodology of PIV namespace management relies on the fact that extended data would be contained in an application that is outside the domain of the NIST PIV Card Application. The application containing extended data would have an AID different from the PIV AID; therefore, the root of the extended data namespace would be different.

An approach to allow extended data inside the PIV application would require some minor namespace management extensions. Specifically, the following structure might be considered:

- Allocate a set of tags that can be used for extended data elements that are unmanaged for inter-industry usage, but contained within a distinct tag range.
- Allocate a set of tags that can be used for extended data elements that are managed specifically by the U.S. Government
- Allocate a set of tags that can be used for extending the PIV namespace for future enhancements that are managed by NIST.

Eventually, it is possible that an independent third party could manage the tags for inter-industry use.

Tables 3-6 and 3-7 show a potential allocation of both OIDs and tags that could be used to extend the PIV data model for NIST, government, and industry usage:

OID Root	Allocation
2.16.840.1.101.3.7.2.xx	NIST PIV OID Namespace
2.16.840.1.101.3.7.3.xx	Inter-agency PIV OID Namespace
2.16.840.1.101.3.7.4.xx	Inter-industry PIV OID Namespace

Table 3-6: Potential OID Namespace Allocation

Tag Root	Allocation
5FC1xx	NIST PIV Tag Namespace
5FC3xx	Inter-agency PIV Tag Namespace
5FC4xx	Inter-industry PIV Tag Namespace

Table 3-7: Potential Tag Namespace Allocation

3.4.4 Creation of New Objects Using Put Data

ISO 7816-4 [3] loosely defines the semantics of using the Put Data command for writing objects onto the card. The statement, “The definition or the nature or the content of the data objects shall induce the exact management functions, e.g., writing once and/or updating and/or appending.” [3] basically says that the security policy applied to data objects determines what operations can be performed on them by specific entities after creation. It does not specifically describe how this security policy is attached to the creation of the object, although one can use the same or similar semantics as those used to create files.

The desired behavior for creating new objects would be to use the existing PIV Put Data command and attach some components of the File Control Parameters (FCP) defined in ISO 7816-4 [3]. The initial strategy might be to use only the core, needed components of the FCP so that access control rules and potentially initial sizes might be attached to a created data object. Given the tag-oriented nature of the FCP, extending the mandatory components should not inhibit the use of a PIV card application that has been extended to realize additional components of the FCP.

3.4.5 PIV API Additions for Object Creation

The SP 800-73/Part 3 Client API uses OIDs for referencing objects. Currently, these OIDs must be mapped to their co-existent tag, which is used on the card for naming data objects. If the PIV namespace is allocated as described previously to allow extensions to the namespace, it will become necessary to provide a mapping mechanism so that new OIDs properly map to new tags when functions like pivPutData are called for the purpose of creating a new on-card data object. One method for meeting this would be to define an OID root for both the government-owned tag scheme and an OID root for inter-industry tag namespace. New objects created under these OID roots would realize their tag by the value appended to the root that was sent through the PIV end state Application Programming Interface (API).

Examples:

```
pivPutData(cardHandle, “2.16.840.1.101.3.7.3.4”,
           dataBuffer);
```

When this PIV Client API function is called, the lower level middleware would recognize that this OID represents the PIV interagency OID namespace based on the root: 2.16.840.1.101.3.7.3. It would take the trailing 4 and append it to the associated tag root namespace for inter-agency use which would be 5FC3. The result would be 5FC304 which would then be passed to the PutData apdu.

```
pivGetData(cardHandle, “2.16.840.1.101.3.7.4.10”,
           dataBuffer);
```

In the above example, the PIV middleware client is requesting to read the data object contained within the interagency OID namespace based upon the root: 2.16.840.1.101.3.7.4. The PIV middleware would recognize this is as residing in the interagency namespace, take the trailing 10, and append it to the associated tag for interagency namespace, which would be 5FC4. The resulting tag for GetData would be 5FC40A. Recall that OIDs are in decimal, and tags are expressed in hexadecimal notation, thus the 0A for the tag instead of 10.

It may be desired to allocate a range of tags and OIDs that can be used for recovered key management keys and certificates. These could be pre-allocated by NIST or the government in their associated namespace or assigned to the inter-industry namespace. If the discovery mechanisms are chosen, which are outlined in this document, these new objects may not need to be allocated in a namespace, but rather discovered.

3.5 Configuring Access Control Rules

Access Control Rules (ACR) are used to describe the access rights to a given set of objects or commands depending on certain procedures that must take place to authorize the requested action. ACRs can be complicated to express depending on how granular they are in terms of users, groups, operations, and data. A simple ACR can be conveyed in a minimal amount of space. For example, traditional UNIX

access control rules are simply bit masks stating whether a user, group, or everyone has read, write, or execute privileges on particular files. Smart cards can have varying complexity of ACRs based on what features the card or card applications provides. Simple memory cards may have no access control or simply the ability to protect data with a user PIN. More complex cards and on card applications may protect operations and data based on multiple authentication levels with various permutations based on AND, OR, and NOT conditions.

To properly support the creation of new objects on the card, there will be a requirement to bind ACRs to the newly created objects. Typically, this is accomplished during object creation and these ACRs can be specified through standard ISO conventions. ISO 7816-4 provides a mechanism for conveying access control rules when creating a file or object on the card.

Numerous options exist regarding ACR format, such as compact format and expanded format. Compact format consists of an access control rule that contains an access mode byte which is followed by one ore more security condition bytes. Specifically, for data objects that are used in SP 800-73, the access mode byte would allow ACRs to be placed on whether the Get and Put data commands could be exercised on a particular data object.

Compact form has limitations in that it cannot effectively be used to express arbitrary Boolean expressions of the security conditions. For example in compact form one could represent a series of AND conditions or a series of OR conditions. They cannot be mixed and one does not have the NOT condition as the expanded format does.

Expanded format contains an access mode data object followed by one or more security condition data objects. Note the distinction in the use of the term object versus byte in expanded and compact format respectively. Expanded format produces more complicated but more flexible conveyance of ACRs. Given this flexibility, expanded format would be a better option for expressing PIV ACRs.

ISO represents access control rules through the File Control Parameters (FCP), which are bound to a file or object during its creation. Using the expanded security attribute template, this information would be contained within the FCP as tag 0xAB. Using PutData to create, manage, and change objects will keep the SP 800-73 command set clear and concise. PutData could now be overloaded to support the following operations:

- Create a new data object with a given set of ACRs contained in the FCP
    - Assumptions: This object does not exist with the given tag name

- Modify an existing data object and potentially its ACRs
    - Assumptions: This object already exists with a given tag name
    - If a new FCP is presented, the ACRs of the existing object are changed as long as its previous ACRs are met prior to this PutData instance.

This addresses creation and modification of data objects and their associated ACRs. To learn about what ACRs are required for particular data objects, the 7816-15 data structure can be referenced and each entry should contain an octet string of ACRs. By simply reading these structures, the middleware and applications using the middleware should be able to not only discover what is on the card but also how it can be accessed.

ISO does not specifically address the assumptions that must be made about access control rules that are not implicitly specified. Developing the wide variety of permutations of ACRs based on not only what can be done and when but also what cannot be done can be quite complicated. For simplicity, PIV might specify that "all operations that are not explicitly allowed within an ACR, are not allowed at all." This fundamental basis would minimize the complexity of PIV ACRs.

3.5.1 Example ACR in Expanded Format

To restrict the access of the Card Holder Fingerprint I by requiring the Card Holder PIV Card Application PIN, the ACR would have the following expanded format:

AB 0B 80 01 01 A4 06 95 01 08 80 01 80

This decomposes into the following subcomponents:

[AB 0B [80 01 01] [A4 06 [95 01 08] [80 01 80]]]

Specifically, each tag, length, value (TLV) breaks down into the following according to ISO 7816-4

[AB 0B

This notes that this is the security attribute template in expanded format.

……..[80 01 01]

This is the Access Mode Byte, which for data objects is 01, for Get Data.

……..……..[A4 06

This is the Control Reference Template for external or user authentication.

…….[95 01 08]

This is the Usage Qualifier, set to 08 for User authentication, knowledge based.

……[80 01 80]]]

This is the Key Reference used to specify which PIN to use. In this example, the reference 80 is used which is the Cardholder PIV Card Application PIN.

ISO 7816-4 also provides the ability to specify which physical interface a particular ACR is associated with. This applies specifically to PIV as some data elements such as the Card Holder Fingerprint I are to be available over the contact interface but not the contactless interface. This is accomplished through the Physical Interface Security Attribute Template:

Physical Interface Security Attribute Template (0xA1) with Tag 0x91

- 0x01—Access by contact interface only
- 0x02—Access by contactless interface only
- 0x03—Access by either contact or contactless interfaces.

Modern smart card operating systems can distinguish the physical interface for which communication is occurring. The recent JavaCard specifications allow this distinction to be discovered by card applications. Card applications can therefore understand ACRs that are based on a particular physical interface. [11]

The FCP also allows other data objects to be contained within it. For example, one might want to specify the initial size of an object when created using Put Data. Note that the initial size does not denote that it is a fixed size object. The object can shrink and grow as data is written to it. Another possible addition is the Life Cycle Status (LCS). The LCS allows the binding of the lifecycle identifier to a particular object on the card. LCS can signify whether an object is in the following states: Creation, Initialization, Operational (activated), Operational (Deactivated), and Terminated.

Table 3-8. Example of Minimum FCP

Tag	Length	Value
80	Variable	Number of initial bytes in object
8A	1	Life-cycle status
AB	Variable	Security Information in expanded format

3.5.2 User Consent

Many electronic ID programs limit the number of privileged operations a user may perform without reauthentication to the card. In 7816-15, this is known as userConsent. Best practice in fraud deterrence requires that users give positive evidence of consent for each use of their PIV Signature Key. PIV cards require the presentation of the PIV PIN each time the Signature Key is used. userConsent is a field in 7816-15, but it must be enforced on the card. Enforcing the userConsent mechanism might use the notion of counter objects in ISO 7816-8 should it be considered necessary.

3.5.3 PIV API Additions for Object Creation with ACR Support

The PIV Client API could be extended to support the creation of new objects with associated ACRs at the API level. This might be accomplished by either adding a new parameter to the existing pivPutData function or by adding a new function such as pivCreateData. Changing the existing pivPutData would entail an existing API modification that would break backwards compatibility with existing PIV end state API implementations. The new parameter would have to be omitted as many times; no ACR would be attached to an object, but it would be simply written. Consequently, it might make sense to establish a new function call that could assume the following structure:

```
status_word pivCreateData(
     IN handle                cardHandle,
     IN string                OID,
     IN sequence of byte      ACRs
);
```

This structure would maintain backwards compatibility with the current SP 800-73 Part 3 API, because nothing has changed, but would extend it to be able to create new, nonexistent data objects and to potentially change ACRs on existing data objects. Note that ISO is working to

standardize an API for smart card application and object management that contains extensive support for discovery of card capabilities and objects. This work is represented in ISO draft 24727, Part 3. The Part 3 API or a profile thereof should be considered for the next generation PIV middleware API once ISO approves the 24727 standard.

4. Card Personalization Through Host APIs

Beyond the SP 800-73 Client API, numerous higher level cryptographic APIs are in wide use with varying degrees of personalization capability. Crypto API and PKCS#11 [11] are commonly used to access cryptographic devices such as smart cards. Such APIs are used to bridge the gap between sets of higher level cryptographic routines to physical devices with cryptographic capabilities.

Both PKCS#11 and Crypto API have limited personalization capability, which will be outlined in this section. Figure 4-1 shows how the host side APIs interact with other routines and components:

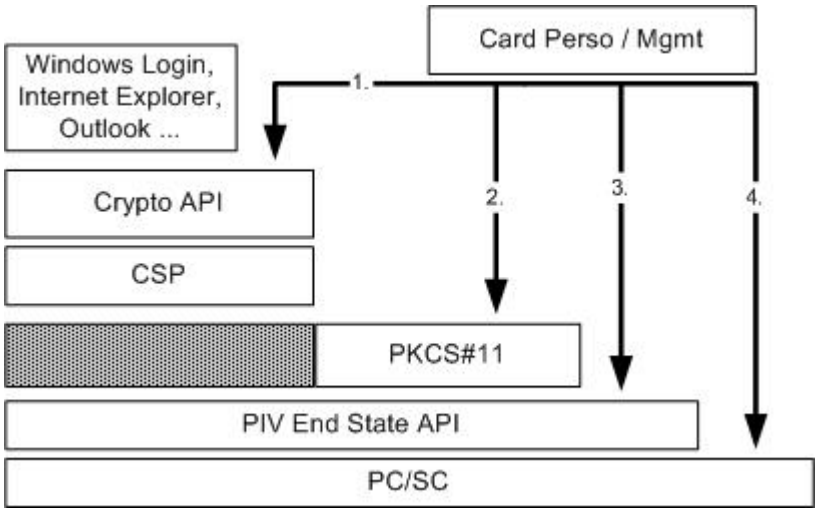

Figure 4-1: Host APIs and Card Personalization

Figure 4-1 shows how a card personalization and management system might gain access to the smart card. Note that the left side of PKCS#11is grayed because some middleware implementations layer their Cryptographic Service Provider(CSP) on top of PKCS#11. The different numbered methods noted in Figure 4-1 are outlined below:

1. **Crypto API.**
    a. Crypto API provides a very limited API for card personalization and management. It supports the creation of basic cryptographic objects such as certificates and keys. It is uncommon for a card personalization and management system to rely solely on crypto API for gaining access to the card.

2. **PKCS#11**
    a. PKCS#11 provides some APIs for card personalization and management. Besides the methods for performing key generation, import, and certificate import, PKCS#11 supports methods for changing/managing PINs and managing generic data objects.
    b. Layering PKCS#11 on top of the PIV Client API entails some complexity when trying to personalize objects that are supposed to be in a particular container. For example, one might wish to generate a key and certificate pair into the PIV authentication slot. The PIV API names objects with OID; PKCS#11 uses CKA_ID. PKCS#11 must be used in a custom manner for generated objects to be created in the correct manner for an implementation that is layered on top of PIV. This could simply be accomplished by

creating objects in a specific, known order. It may also be accomplished by passing a custom attribute to C_CreateObject with the OID to be passed to pivPutData. These complexities are attributed to the nature of the fixed data model that PIV uses.

3. **PIV End State API**
    a. The PIV Client API provides some mechanisms for doing card personalization, specifically to write existing data objects and to generate keys and change default PINs on the card. Currently, no CMSs use this API for abstract card management and personalization.

4. **Personal Computer Smart Card**
    a. Personal Computer Smart Card (PC/SC) is commonly used to communicate directly to the card from the back-end CMS. It has the most control of the card but also the least amount of card abstraction depending on how it is used. PC/SC does contain a card abstraction layer, which may find more use in upcoming versions of Microsoft operating systems.

Each of these lacks an abstract means to perform card application management with ISO 7816-13 or Global Platform. Consequently, card application management is often performed with PC/SC communicating directly with the card over a Secure Sockets Layer (SSL) connection from a CMS server issuing commands to the card. Often, the CMS will use a hybrid combination of PC/SC and PKCS#11 to meet all the requirements to issue and manage cards securely. Standard APIs and operating systems provide reasonable support for the usage of already managed and personalized identity tokens. It is expected that the abstract management and personalization of identity tokens will be addressed in future revisions of standard APIs and operating systems.

5. Summary

The PIV suite of standards and specifications define a foundation for interoperability of smart card identity tokens across federal agencies. Many organizations are investigating the need to augment the card management and personalization capabilities of the PIV suite. By extending the existing PIV card specifications in the card management area, it is possible to maintain backwards compatibility and enable development of fully functional and interoperable PIV cards that can be deployed across multiple CMSs and host middleware.

ISO 7816 defines a toolbox of available commands that can be used for creating interoperable smart card interfaces. These standards have been adopted worldwide in numerous national identity card programs and private sector deployments. SP 800-73 is based on some of the available commands contained within ISO 7816. Expanding the PIV command set to include management and personalization would result in a higher level of consistency and testability for PIV card issuance processes, enhanced ability to outsource various card management components and functions, and improved overall security for the federal PIV framework.

6. Example 7816-15 Composition

Potential 7816-15 Minimum Mandatory Structure

The following depicts a potential minimum, mandatory 7816-15 structures that could reside on the PIV card for discovery purposes.

AOD:

{

Label String

ACR Octet String

AuthReference Integer

PIN:

{

PasswordFlags Integer

MinimumLength Integer

MaximumLength Integer

StoredLength Integer

PwdReference Integer

}

Biometric:

{

BiometricFlags Integer

TemplateOID Octet String

Hand Integer

Finger Integer

}

External Authentication:

{

DerivedKey Integer

Identifier Octet String

}

}

PRKDF:

{

Label	String
ACR	Octet String
Identifier	Octet String
KeyUsageFlags	Integer
Native	Integer
KeyReference	Integer
KeyAccessFlags	Integer
ModulusLength	Integer
userConsent	Integer

}

PUKDF:

{

Label	String
ACR	Octet String
Identifier	Octet String
KeyUsageFlags	Integer
KeyReference	Integer
KeyAccessFlags	Integer
Value	Octet String
Modulus	Octet String

}

CD:

{

Label	String
ACR	Octet String
Identifier	Octet String
Value	Octet String

}

SKD:

{

Label	String
ACR	Octet String
Identifier	Octet String
KeyUsageFlags	Integer
KeyReference	Integer
KeyAccessFlags	Integer
KeyLength	Integer
Algorithms	Octet String

}

Note: ACR can be further expanded into a sequence of Access Modes and Security Conditions as defined in 7816-15. These definitions above are flattened versions of what might contain more hierarchy in structure.

7. List of Acronyms

ACR	Access Control Rule
AID	Application Identifier
AOD	Authentication Objects
API	Application Programming Interface
CCC	Card Capabilities Container
CHV	Card Holder Verification
CMS	Card Management System
FCP	File Control Parameters
GSC-IS	Government Smart Card Interoperability Specification
HSM	Hardware Security Module
HSPD	Homeland Security Presidential Directive
ISO	International Organization for Standardization
ITL	Information Technology Laboratory
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Interagency Report
OID	Object Identifier
PACS	Physical Access Control System
PC/SC	Personal Computer Smart Card
PIN	Personal Identification Number
PIV	Personal Identity Verification
PKCS	Public-Key Cryptography Standards
PUK	PIN Unlock Key
RSA	Rivest, Shamir, and Adleman
SP	Special Publication
SSL	Secure Sockets Layer
TLV	Tag, Length, Value

8. References

1. FIPS Publication 201, Federal Personal Identity Verification (PIV) for Federal Employees and Contractors
http://csrc.nist.gov/publications/fips/fips201/FIPS-201-022505.pdf

2. NIST Special Publication 800-73, Interfaces for Personal Identity Verification
http://csrc.nist.gov/publications/nistpubs/800-73/SP800-73-Final.pdf

3. ISO 7816 (4, 13, 15): Information Technology—Identification Cards—Integrated Circuit Cards with Contacts.
http://www.iso.org

4. FINEID—S4-1 Implementation Profile 1 v 2.1A
http://www.fineid.fi/vrk/fineid/files.nsf/files/E99479C66998368DC2257054002A8686/$file/S4-1v21A.pdf

5. FINEID S1 Electronic ID Application v 2.1
http://www.fineid.fi/vrk/fineid/files.nsf/files/4A6480742C01D98BC2257054002A1D23/$file/S1v21.pdf

6. Belgian Electronic Identity Card Content v 2.8a
http://www.rijksregister.fgov.be/cie/specifications_techniques/belgian_electronic_identity_card_content_v2.8.a.pdf

7. German Health Card Specifications
http://www.dimdi.de/static/de/ehealth/karte/download/egk_spezifikation_teil1_v1-2.pdf
http://www.dimdi.de/static/de/ehealth/karte/download/egk_spezifikation_teil2_v1-0.pdf

8. Global Platform Card Specification Version 2.1.1
http://www.globalplatform.org

9. Government Smart Card Interoperability Specification Version 2.1
http://csrc.nist.gov/publications/nistir/nistir-6887.pdf

10. PKCS #15: Cryptographic Token Information Format Standard
http://www.rsasecurity.com/rsalabs/pkcs

11. Java Card Platform Specification v2.2.1

http://java.sun.com/products/javacard/specs.html