PK-PACS Reader Specification

Taglio, LLC

August 10, 2023

Contents

Abstract

The PK-PACS Specification defines a method and data model for the support of Public Key based card authentication compatible with any existing Physical Access Control System.

The PK-PACS Specification defines how to store legacy identifiers (such as the facility and card number used by Proximity cards) into a digital certificate on a smart card. A PK-PACS compatible door reader can authenticate the card and verify the certificate with Public Key cryptography. The reader can then retrieve the legacy identifier from the card certificate and return it to the any Physical Access Control System without any changes to that system.

The PK-PACS Specification aims to ensure interoperability between Public Key enabled cards and readers, as well as readers and Physical Access Control Systems.

Introduction

Smart Cards are hardware security devices that are used to authenticate users and devices, and to enforce security policies on the client platforms. Smart cards that support Public Key cryptography are widely used today. Most Credit and Debit smart cards are Public Key capable, as well as SIM cards for Mobile Phones, and national identity cards. Many organizations use Smart Cards with Public Key cryptography for logon to computers and many other applications. However, one area where Smart Cards with Public Key cryptography are not commonly used is Physical Access. Organizations that have issued smart cards to users for computer logon cannot easily use these cards for physical access. Current systems that support Public Key cryptography for Physical Access are available, but are perceived as complex and expensive, requiring the replacement of not just readers and cards, but also the Physical Access Control hardware, and a change in processes for issuing and enrolling cards. The PK-PACS specification simplifies the use of Smart Cards for Physical Access. PK-PACS enabled cards and readers can support any legacy Physical Access system.

Scope of this Document

This document specifies how a physical access Reader should read a credential from a PK-PACS enabled Contactless Smart card using the PIV Card Application Interface.

Definitions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”,“SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Card: A contactless card or dual interface card with an application that supports the PIV Card Application interface and has a X.509 Certificate for Card Authentication

Reader: Electronic device, trusted by the PACS, that is used to authenticate the Card.

PACS: Physical Access Control System, including components such controller, lock, cabling and backend system.

ACL: (Access Control List) A list of PACS identifiers used by the PACS to determine which Card is allowed access.

PIV Card Application Interface: Card API defined by NIST.SP.800-73-4, Part 2: PIV Card Application Card Command Interface.

PK-Cert: X.509 Certificate for Card Authentication retrieved from PIV Card Application Interface used for authentication to the PACS Reader.

PK-PubKey: The Public Key in the PK-Cert that is used to verify the Card

PK-PrivKey: The Private key in the Card associated with PK-PubKey

PACS Identifier: A data element in the PK-Cert that can be used to identify the card to the PACS system. This can be a UUID, 4/7/10 Byte Identifier, or FAC/CSN identifier.

PACS Reader Output Format (Output Format): The data format the Reader uses to communicate the PACS Identifier to the PACS system. This can be a legacy format such as a 24 125 kHz Prox

PK-TrustKey – Trusted Public Key and associated key information that is used to verify the PK-Cert Signature to determine whether the PK-Cert content is trusted.

PK-TrustCert – Certificate with PK-TrustKey

ID-OID: A X.509 OID that identifies the Data Element that contains the PACS Identifier.

PK-PACS ARC: OID ARC with the value '1.3.6.1.4.1.59685', representing the IANA Private Enterprise Number PK-PACS.

ID-OID and PACS Identifiers

Legacy PACS cards support a variety of legacy card identifier types. Low Frequency cards typically use an identifier consisting of a series of bits representing a Card Serial Number and Facility Code. High Frequency cards may provide a UID.

PK-PACS cards represent these legacy card identifier types as X.509 data elements in the PK-CERT. Card identifiers, including the legacy identifiers are identified in the certificate through an X.509 Object Identifier (OID) known as an ID-OID. PK-PACS Readers support one or more PACS identifier that it can accept.

ID-OIDs are based on IANA Private Enterprise Numbers. The default PK-PACS ID-OIDs are defined on the PK-PACS Arc with the value '1.3.6.1.4.1.59685'. Other Cards Issuers may support other registered Private Enterprise Numbers.

By convention ID-OIDs are referenced by a shortened OID. The Full OID can be constructed from the ID-OID by prepending "1.3.6.1.4.1". For example the OID for ID-OID 59685.8.1 is 1.3.6.1.4.1.59685.8.1.

The following PK-PACS Identifiers and ID-OIDS are specified:

Table 1: PK-PACS ID-OIDS

Identifer Name	ID-OID	Example/X.509 Data
UUID	59685.8.1	OCTET STRING (18 byte) 04104705EFDE35234512BF65DE446AFB5943
4 Byte NUID	59685.8.2	OCTET STRING (6 byte) 0404EA9963AC
7/10 Byte UID	59685.8.3	OCTET STRING (9 byte) 0407EA9963AC9963AC
FAC/CSN	59685.8.8	UTF8String 00001000000022

UUID/NUIDs

UIDs are numbers assigned by the card issuers that are not Serial Numbers. They can be universally unique or not. PK-PACS supports common UIDs, including the 4/7/10 byte UID format defined by the ISO 14443, and supported by the proprietary Mifare(R) and DESFire (R) line of proucts from NXP (R), as well as the RFC 4122 type GUID commonly used in IT applications and supported by the PIV Card Application.

FAC/CSN Identifier

Many Legacy Identifiers used by 125KhZ Proximity cards contain both a Facility number and Card Serial Number. The legacy formatting is typically done in a series of bits. Many different types of formats are available, but most can be programmed to support a FAC and CSN.

The FAC/CSN Identifier stores a facility code and card serial number in a human readable format that the reader can then convert into a specific legacy output format.

The FAC/CSN Identifier is a string of 14 numeric characters. The first 7 characters represent the Facility Code, the last 7 characters represent the Card Serial number. For example, the FAC/CSN Identifier “00001000000022” represents an identifier with Facility Number 100 and Card Serial Number 22.

Requirements

Reader MUST support all ID-OIDS in table 1.

Reader MAY support ID-OIDs that are defined on a registered IANA Private Enterprise Number Arcs.

Reader MUST NOT support ID-OIDs that are not defined on a registered IANA Private Enterprise Number Arcs.

Reader MAY be configured to accept one or more ID-OIDS for a specific implementation

Reader MAY implement application logic to determine which ID-OID will be used if multiple ID-OIDs are available on the card.

PACS Reader Output Formats

The Reader retrieves the PACS Identifier from the data element identified with the ID-OID. The Reader then converts the PACS Identifier to an Output Format that can be processed by a Reader compatible PACS.

For example, the Reader may convert the FAC/CSN to a 26-bit (Wiegand H10301) identifier recognized by the PACS system. The exact output format

will vary according to the specific requirements of the PACS.

Table 2 Output Formats

Code	Identifier Name	Output Format
UF01	UUID	4705efde-3523-4512-bf65-de446afb5943
UF02	4 Byte NUID	ea 99 64 ac
UF03	7/10 Byte UID	ea 99 63 ac 99 63 ac
UF08	FAC/CSN	00001000000022

Requirements

Reader MUST support one or more output formats for each entry listed in Table 2.

Card Authentication Use Case

The following use case describes the Card Holder authenticating to a Reader to open a door. A precondition is that the cardholder has been added to the ACL.

1. Cardholder presents Card to Reader
2. Reader reads PK-Cert from Card
3. Reader Parses PK-Cert to retrieve PK-PubKey and PACS Identifier(s) identified with ID-OID from the ID-OID list
4. Reader determines PACS Identifier to use
5. Reader determines PK-TrustKey and Output Format associated with the selected PACS Identifier.
6. Reader sends cryptographic challenge to Card
7. Card generates cryptographic response to challenge using PK-PrivKey
8. Reader retrieves cryptographic response from Card
9. Reader verifies cryptographic response
10. Reader verifies signature on PK-Cert using PK-TrustKey
11. Reader sends card identifier to PACS
12. PACS checks ACL
13. PACS opens door

Card Authentication Requirements

Reader MUST support ISO/IEC 14443 to communicate with the Card.

Reader MUST read the PK-Cert from the Card

Reader MUST decompress a compressed PK-Cert

Reader MUST be able to parse and retrieve the data elements from PK-Cert

Reader MUST be able to execute an External Authenticate Command Challenge according to NIST.SP.800-73-4 Part 2, using the 9E Key on the card, and retrieve the Response

Reader MUST be able to verify the Response using a PK-PubKey

Reader MUST be able to verify the cryptographic signature on PK-Cert using PK-TrustKey

Output Communications Protocols

Output communication protocols are used to communicate between the Reader and the PACS.

Reader MUST support an Output Method compatible with the PACS system(s) for which it is designed.

Reader MAY support Wiegand

Reader MAY support OSDP

Reader MAY support a REST API

Reader MAY support USB

Reader MAY support a proprietary PACS API

PK-TrustKey Trusted Signing Key

The PK-TrustKey is a public key that is used to verify the signature on the PK-Cert, and assure that the data elements in the cert are not tampered with. The PK-TrustKey may be part of a trust chain, and may be embedded in a signing certificate.

Reader MUST support TWO or more PK-TrustKeys

Reader MUST verify certificate signatures signed with a RSA 2048 PK-Trustkey

Reader MAY verify certificate signatures signed with a RSA 4096 PK-Trustkey

Reader MAY verify certificate signatures signed with a PK-Trustkey based on any other Cryptographic Algorithm supported by NIST.SP.800-73-4 Part 2.

Reader MAY store the PK-TrustKey as part of a Certificate, such as a Root or Subsidiary Certificate.

Reader MAY store only the TrustKey itself and any relevant data elements from the certificate needed to execute the required cryptography on the Reader.

Reader MUST NOT verify the PK-TrustKey trust chain as part of the authentication process.

Reader MUST associate the Card and/or PK-Cert with a PK-TrustKey to verify the signature on the PK-Cert

PK-PubKey/PK-PrivKey

The PK-PubKey/PK-PrivKey are the key pair that is used to authenticate the card.

The Reader MUST support a PK-PubKey based on RSA 2048

The Reader MAY support a PK-PubKey pair based on any other Cryptographic Algorithm supported by NIST.SP.800-73-4.

Reader Configuration

Readers can be configured to support different ID-OIDs, PK-Trustkeys and output methods.

The Reader MUST have a process for updating the ID-OIDs supported on the Reader

The Reader MUST have a process for updating the PK-Trustkeys on the Reader

The Reader MAY have a process for updating the output formats on the Reader

If the Output Communication Protocol is OSDP, then the reader MUST support a process for updating using OSDP commands

Implementation Guidance

A: Card Application Interface

The challenge uses the External Authenticate Command against the 9E Key on the card. See NIST.SP.800-73-4 Part 2, page 40, A.3 Authentication of PIV Cardholder

Example PK-PACS Certificate

PEM Format

```
-----BEGIN CERTIFICATE-----
MIIEEjCCAvqgAwIBAgIUUD4PMpccpxmNhO/ZzprPYStO818wDQYJKoZIhvcNAQEL
BQAwSjEpMCcGA1UEAwwgVGFnbGlvIERlbW9uc3RyYXRpb24gRGV2aWNlIENBIDEx
EDAOBgNVBAoMB1NlY3VwYXMxCzAJBgNVBAYTAkVVMB4XDTIzMDIyMTE0MTE0M1oX
DTMwMDQxMDE4NTgzN1owLzEtMCsGA1UEAwwkMGMzNGZhYTMtMTA5OC00YTgxLWI2
YjItMzJhMzk1ZWU5YzJlMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
voKotRYPul3qJeCCugNIWSRG2oHKsc0X4gh1p5hInjYfLU8muViQczlYg2gDoywW
eCUSUHwnZheppGtqaSKoFd+JpJqMJCYLTXYBjVcXErUU/mZE4K2SaCX+yhpZowsM
EKdFeekl3NaIZZ19ubrwzYL9hocgfKKUIzAKKvHiaztaC4DeC9HRwZnKYEf/r8e3
W3B0vEEx+9VF6Rk3/81qzkimPf6RE3pFxxI3hm46gipakLRWHBkCie9mRqUWh03l
f6zxAobzojlzBMtRSccAw/Lfc6wSDsUFItmItbhTHxHtSZKQ5TrRL2Ls6gOEm1EC
K20bkMllkdw4WqNAAGsJ6QIDAQABo4IBCTCCAQUwHwYDVR0jBBgwFoAUOpoGcCr2
Owmmkkavne7cOY+RE7QwEwYDVR0gBAwwCjAIBgYEAI96AQMwMgYDVR0lBCswKQYI
KwYBBQUHAwIGBysGAQUCAwQGCisGAQQBgjcUAgIGCCsGAQUFBwMVMB0GA1UdDgQW
BBRUTejEmYRr2tVZIsvAahxH1WdYHzAOBgNVHQ8BAf8EBAMCB4AwEgYLKwYBBAGC
3zoCBQAEAwEB/zAeBgorBgEEAYLfOggIBBAMDjAwMDAxMDAwMDAwMDAxMBQGCisG
AQQBgt86CAIEBgQEandjrDAgBgorBgEEAYLfOggBBBIEEAw0+qMQmEqBtrIyo5Xu
nC4wDQYJKoZIhvcNAQELBQADggEBAFxTkJWiW78crjMapnIYhw2fmlluFjIOcM1A
sOLrTs7XDPZ9lXHTDTF8SUD8Oij/AEvndPbkhVSIf9FFxC4s/J6pbVuLc5dlJkKa
3qWQ9BcvGiWXTK3vQJhS3dQYy82NzUScuujdPIhgfFHT7xAWju+ZmRkWHjttEwUE
gyYmRgtPv28Qe05LzJ9b/rC7kqiAIHV7KYahm25BHcHWJv6je8jzy6yleyH4hD6I
akvDedSQbXteJ7BjV6JOtGoIV8C9ikU86qUrk8Qgc4EPqt/0tFOtpbsPbdcVndZz
M1LCnxFM68hjaTld4ElTN/uV164q4ZXRi7bDJeR2tQbmozEt3mQ=
-----END CERTIFICATE-----
```

ASN.1 JavaScript decoder

```
SEQUENCE (3 elem)
  SEQUENCE (8 elem)
    [0] (1 elem)
      INTEGER 2
    INTEGER (159 bit) 458103231767485583886516839208027605779212071775
    SEQUENCE (2 elem)
      OBJECT IDENTIFIER 1.2.840.113549.1.1.11 sha256WithRSAEncryption (PKCS #1)
      NULL
    SEQUENCE (3 elem)
      SET (1 elem)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.5.4.3 commonName (X.520 DN component)
          UTF8String Taglio Demonstration Device CA 1
      SET (1 elem)
        SEQUENCE (2 elem)
```

```
      OBJECT IDENTIFIER 2.5.4.10 organizationName (X.520 DN component)
      UTF8String Secupas
  SET (1 elem)
    SEQUENCE (2 elem)
      OBJECT IDENTIFIER 2.5.4.6 countryName (X.520 DN component)
      PrintableString EU
SEQUENCE (2 elem)
  UTCTime 2023-02-21 14:11:43 UTC
  UTCTime 2030-04-10 18:58:37 UTC
SEQUENCE (1 elem)
  SET (1 elem)
    SEQUENCE (2 elem)
      OBJECT IDENTIFIER 2.5.4.3 commonName (X.520 DN component)
      UTF8String 0c34faa3-1098-4a81-b6b2-32a395ee9c2e
SEQUENCE (2 elem)
  SEQUENCE (2 elem)
    OBJECT IDENTIFIER 1.2.840.113549.1.1.1 rsaEncryption (PKCS #1)
    NULL
  BIT STRING (2160 bit) 001100001000001000000001000010100000001010...
    SEQUENCE (2 elem)
      INTEGER (2048 bit) 240497083042906858242113933238517526746 48...
      INTEGER 65537
[3] (1 elem)
  SEQUENCE (9 elem)
    SEQUENCE (2 elem)
      OBJECT IDENTIFIER 2.5.29.35 authorityKeyIdentifier
        (X.509 extension)
      OCTET STRING (24 byte)
        301680143A9A06702AF63B09A69246AF9DEEDC398F9113B4
        SEQUENCE (1 elem)
          [0] (20 byte) 3A9A06702AF63B09A69246AF9DEEDC398F9113B4
    SEQUENCE (2 elem)
      OBJECT IDENTIFIER 2.5.29.32 certificatePolicies (X.509 extension)
      OCTET STRING (12 byte) 300A3008060604008F7A0103
        SEQUENCE (1 elem)
          SEQUENCE (1 elem)
            OBJECT IDENTIFIER 0.4.0.2042.1.3 lightweightCertificatePolicy
              (ETSI TS 102 042 Certificate Policies)
    SEQUENCE (2 elem)
      OBJECT IDENTIFIER 2.5.29.37 extKeyUsage (X.509 extension)
      OCTET STRING (43 byte) 302906082B06010505070302060 72B06010502...
        SEQUENCE (4 elem)
          OBJECT IDENTIFIER 1.3.6.1.5.5.7.3.2 clientAuth (PKIX key purpose)
          OBJECT IDENTIFIER 1.3.6.1.5.2.3.4 keyPurposeClientAuth (Kerberos)
          OBJECT IDENTIFIER 1.3.6.1.4.1.311.20.2.2 smartcardLogon
            (Microsoft extended key usage)
```

```
            OBJECT IDENTIFIER 1.3.6.1.5.5.7.3.21 secureShellClient
              (PKIX key purpose)
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 2.5.29.14 subjectKeyIdentifier (X.509 extension)
          OCTET STRING (22 byte) 0414544DE8C499846BDAD55922CBC06A1C47D567581F
            OCTET STRING (20 byte) 544DE8C499846BDAD55922CBC06A1C47D567581F
        SEQUENCE (3 elem)
          OBJECT IDENTIFIER 2.5.29.15 keyUsage (X.509 extension)
          BOOLEAN true
          OCTET STRING (4 byte) 03020780
            BIT STRING (1 bit) 1
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 1.3.6.1.4.1.44986.2.5.0
          OCTET STRING (3 byte) 0101FF
            BOOLEAN true
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 1.3.6.1.4.1.44986.8.8
          OCTET STRING (16 byte) 0C0E30303030313030303030303031
            UTF8String 00001000000001
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 1.3.6.1.4.1.44986.8.2
          OCTET STRING (6 byte) 04046A7763AC
            OCTET STRING (4 byte) 6A7763AC
        SEQUENCE (2 elem)
          OBJECT IDENTIFIER 1.3.6.1.4.1.44986.8.1
          OCTET STRING (18 byte) 04100C34FAA310984A81B6B232A395EE9C2E
            OCTET STRING (16 byte) 0C34FAA310984A81B6B232A395EE9C2E
SEQUENCE (2 elem)
  OBJECT IDENTIFIER 1.2.840.113549.1.1.11 sha256WithRSAEncryption (PKCS #1)
  NULL
BIT STRING (2048 bit) 010111000101001110010000100101O...
```