PDF Public-Key Digital Signature and Encryption Specification

Version 3.2, September 12, 2001 3:35 pm
Jim Pravetz, Acrobat Engineering
(jdp019)

This document was previously referred to as *PDF Private Data Specification for Public-Private Key (PPK) Digital Signatures and Secure Documents*.

1 Overview

This specification describes a standard syntax for the storage of data in the *signature dictionary and* the *encryption dictionary* of plug-ins that use *public-key* technology. A degree of interoperability of signed and encrypted PDF files across plug-ins and over time should be possible by using a standardized syntax (see Signature and Encryption Handler Interoperability).

This document corresponds with the formats used in Acrobat 4.0 through 5.0.

1.1 Signature and Encryption Handler Interoperability

It is intended that there be interoperability between signature and encryption handlers. This would allow PDF files authored with a plug-in from one vendor can be processed by a recipient who is using a plug-in from a different vendor. An example where this is useful is when a document signed using a plug-in that supports an Entrust infrastructure is being authenticated by a recipient who is using a plug-in that supports a VeriSign infrastructure. Another example is a document that is encrypted with Entrust being decrypted by the Self-Sign plug-in.

The interoperability model as we saw it in the Acrobat 4.0 time frame used the **Filter** and **SubFilter** attributes of the signature or encryption dictionaries to achieve interoperability. The **Filter** attribute of the signature or encryption dictionary indicated the handler that was to be used to process the dictionary contents. The **SubFilter** attribute indicated the standardized syntax (format) of the dictionary contents. The plug-in that corresponded to the value of the **Filter** attribute was the *primary* handler that was used to process the dictionary contents. If this handler was not present (had not registered) then the value of the **SubFilter** attribute was used to find an *alternate* handler that could be used. Each registered handler would be called, in an unspecified order, until one was encountered that was able to parse the format indicated by the **SubFilter** value.

Moving forward after Acrobat 5.0 we may refine the mechanism used to provide interoperability. Once this mechanism is defined we will update this specification.

2 Signature Dictionary

The signature dictionary is the value of the **V** attribute of the *signature field dictionary*. The signature field dictionary and many of the attributes in the signature dictionary are described in the PDF Reference Manual 1.3 under Digital Signatures.

There are two alternate formats for storing public key signatures in the signature dictionary of a PDF file. One, the Raw Signature Format, stores certificates and the signed digest directly in the signature dictionary as attributes. The other, the PKCS#7 Signature Format, encapsulates the signature and the signed digest into a single PKCS#7 object that is stored in the dictionary. There are advantages and disadvantages to both formats.

An advantage of PKCS#7 is it is a format that can be parsed and generated directly by most high level public-key toolkits. These toolkits often do not expose the routines that are required to generate the Raw Signature Format. A disadvantage of using PKCS#7 is that 5 to 10 Kilobytes of space are required for the PKCS#7 object (the size is doubled because it is hex encoded).

An advantage of the Raw Signature Format is that the signature is more compact. Only the signed digest must be hex encoded, meaning that X.509 certificates can be represented more compactly. The PKCS#7 object also contains redundant information that is not required. This redundant information includes: the ability to have multiple signers in one signature (multiple signers are supported in PDF by a serial mechanism); signer information that includes the signing time; and replication of algorithm information that is implicit in the **SubFilter** attribute.

The PKCS#7 object is simply a wrapper for signing information that can, in the case of the Raw Signature Format, be expressed directly as attributes in the signature dictionary. It is possible to build a PKCS#7 object on-the-fly from the contents of the signature dictionary used for the Raw Signature Format. There is therefore no disadvantage to using the Raw Signature Format where it is supported by a toolkit. Signature handlers should be able to verify both formats. Upon request Adobe will make available to developers the wrapper code that builds a PKCS#7 object from the Raw Signature Format.

2.1 PKCS#7 Signature Format

This syntax use a PKCS#7 object to encapsulate the document signature and signer’s X.509 certificates. The **SubFilter** value should be **adbe.pkcs7.detached**.

The first nine entries in this table, entries **Type** through **Contents**, are standard keys that are defined in the *PDF Reference Manual 1.3* under *Signature Dictionary Attributes*. **V** is also a standard key but has not previously been documented because it is not yet used (it defaults to 0).

Table 1 *Digital Signature Dictionary Attributes for PKCS#7 Format*

Key	*Type*	*Semantics*
Type	name	(*Required*) The type of the signature dictionary. Always **Sig**.
Filter	name	(*Required*) Language independent name of the digital signature handler.
SubFilter	name	(*Required, if it is desired that other registered security handlers be able to authenticate the signature*) Either **adbe.pkcs7.detached** or **adbe.pkcs7.sha1**. This is the name describing the format of data contained in the signature dictionary. These formats require that the PKCS#7 object be stored in the **Contents** attribute of the signature dictionary. *Note: For optimal interoperability it is recommended that the PKCS#7 format* **adbe.pkcs7.detached** *be used. The format* **adbe.pkcs7.sha1** *was an early PKCS#7 format that is only supported in a select number of signature handler revisions.*
Name	string	(*Required*) Contains a common name by which the signer of the PDF file can be referred.
Reason	string	(*Optional*) A reason why the signer is signing the PDF file. The signer might, for example, give a reason of *approved for distribution*.
Location	string	(*Optional*) String giving the CPU hostname or physical location of the signing. For example, *San Jose, CA, USA*.
M (Mod Date)	date	(*Required*) The date and time at which the signature was applied to the document. This is the modification date for the signature dictionary based on the machine date of the machine used for signing. This is not a date that is verifiable by, for example, a timestamp authority.
ByteRange	string	(*Required*) An array of two pairs of integers, describing the exact byte range for the digest calculation: start byte-offset, length in bytes. This will include all bytes in the PDF file with the exclusion of the bytes represented by the value of the **Contents** attribute. The bytes excluded include the hex string delimiters **<** and **>** of the **Contents** value.
Contents	hex string	(*Required*) A binary PKCS#7 object containing signed data and represented as a hex string. The data encapsulated into this PKCS#7 object (signed-data) is either (i) empty, if **SubFilter** is **adbe.pkcs7.detached**, or (ii) the SHA-1 hash of the bytes specified by the **ByteRange** attribute, if **SubFilter** is **adbe.pkcs7.sha1**. The minimum required certificate that is included in this object is the signer's X.509 verification (signing) certificate. The object may optionally contain one or more issuer certificates in the signer's certificate trust chain. The value must be a hex string with the **<** and **>** delimiters fitting precisely within the hole defined by **ByteRange**. There may be padding following the octet string because the value is placed in the document using a dummy value and then overwritten after the document is saved. If the length of the contents can be variable it will be necessary to pad the value: padding must use zeros at the end of the string, but before the **>** delimiter.

V (Version)	integer	(*Reserved*) Indicates the version of the signature dictionary format, which corresponds with the usage of the signature dictionary as defined in the PDF Reference Manual version 1.3. Currently not used. Defaults to 0.
R (Revision)	integer	(*Optional*) Specifies the version of the digital signature handler that is being used. This value is specific to the digital signature handler.
ADBE_Build	string	(*Optional*) Introduced in Acrobat 4.05. Indicates the precise build of software that was used to create this signature, giving information such as platform and build date. This string is not intended to be machine readable. Rather it is intended to be used in claims of repudiation.
ADBE_AuthType	name	(*Optional*) Specifies the method that the signer used to authenticate themselves with the plug-in. If not present then the authentication method is considered to be by password entry (indicated by a value of **/Password**). The only other possible value that is currently defined is **/Fingerprint**. If other authentication types are required then these should be reported back to Adobe so they can be added to this specification. Intended to be used in claims of repudiation.
ADBE_PwdTime	number	(*Optional*) The number of seconds since the last time the signer was required to authenticate themselves using, for example, a keyed-in password. Intended to be used in claims of repudiation.

2.2 Raw Signature Format

The raw signature format stores the signed digest and the signer's certificates directly as attributes in the signature dictionary. This is different from the PKCS#7 Signature Format which encapsulates this information in a PKCS#7 object. Currently the only defined raw signature format is **adbe.x509.rsa_sha1**, which uses RSA private key encryption and the SHA-1 message digest algorithm. The message digest algorithm is also defined as an encoded attribute in the **Contents** string, however it is replicated in the SubFilter key to make it more accessible. You can conclude that other **SubFilter** values that might be legal (but not yet implemented) are, for example, **adbe.x509.dsa_sha1** and **adbe.x509.rsa_md5**.

The first nine entries in this table, entries **Type** through **Contents**, are standard keys that are defined in the *PDF Reference Manual 1.3* under *Signature Dictionary Attributes*. **V** is also a standard key but has not previously been documented because it is not yet used.

Table 2 *Digital Signature Dictionary Attributes for X.509 Format*

Key	*Type*	*Semantics*
Type	name	(*Required*) The type of the signature dictionary. Always **Sig**.
Filter	name	(*Required*) Language independent name of the digital signature handler. The name of a demonstration handler from Adobe is **Adobe.PPKLite**.

SubFilter	name	(*Required, if it is desired that other registered security handlers be able to authenticate the signature*) A name describing the format of data contained in the signature dictionary. Always **adbe.x509.rsa_sha1**. The name **adbe.x509.rsa_sha1** indicates an Adobe-originated format based on X.509 certificates, the RSA encryption algorithm, and the SHA-1 message digest algorithm.
Name	string	(*Required*) Contains a common name by which the signer of the PDF file can be referred.
Reason	string	(*Optional*) A reason why the signer is signing the PDF file. The signer might, for example, give a reason of *approved for distribution*.
Location	string	(*Optional*) String giving the CPU hostname or physical location of the signing. For example, *San Jose, CA, USA*.
M (Mod Date)	date	(*Required*) The date and time at which the signature was applied to the document. This is the modification date for the signature dictionary based on the machine date of the machine used for signing. This is not a date that is verifiable by, for example, a timestamp authority.
ByteRange	string	(*Optional*) An array of two pairs of integers, describing the exact byte range for the digest calculation: start byte-offset, length in bytes. This will include all bytes in the PDF file with the exclusion of the bytes represented by the value of the **Cert** attribute. The bytes excluded include the hex string delimiters **<** and **>** of the **Cert** value.
Contents	hex string	(*Required*) The private-key encrypted message digest of the bytes defined in **ByteRange**. The message digest is signed using the private key that matches the public key contained in the first (or only) X.509 certificate in **Cert**. For the **SubFilter** format defined by **adbe.x509.rsa_sha1** the private key is an RSA private key. The value must be a hex string with the **<** and **>** delimiters fitting precisely within the hole defined by **ByteRange**. Inside this value is a BER encoded octet string. Inside this BER encoded octet string is an RSA encrypted BER encoded message digest. Specifically the message digest is encoded as a sequence with the first entry in the sequence being the OID of the message digest algorithm, and the second entry being an octet string containing the message digest. The OID identifies the message digest algorithm and should be SHA-1 if the **SubFilter** value ends in sha1. There may be padding following the octet string because the value is placed in the document using a dummy value and then overwritten after the document is saved. If the length of the contents can be variable it will be necessary to pad the value: padding must use zeros at the end of the string, but before the **>** delimiter.
Cert	string or array	(*Required*) The X.509 public key certificate of the signing party, or an array of X.509 certificates. When an array, the first entry in the array must be the X.509 certificate of the signer. Only one signer is permitted per signature dictionary. Subsequent entries in the array must then be an ordered list of parent X.509 certificates in the certificate trust hierarchy, with the topmost certificate in the hierarchy appearing last.

R (Revision)	integer	(*Optional*) Specifies the version of the digital signature handler that is being used. This value is specific and private to the digital signature handler.
V (Version)	integer	(*Reserved*) Indicates the version of the signature dictionary format, which corresponds with the usage of the signature dictionary as defined in the PDF Reference Manual version 1.3. Currently not used. Defaults to 0.
ADBE_Build	string	(*Optional*) Introduced in Acrobat 4.05. Indicates the precise build of software that was used to create this signature, giving information such as platform and build date. This string is not intended to be machine readable. Rather it is intended to be used in claims of repudiation.
ADBE_AuthType	name	(*Optional*) Specifies the method that the signer used to authenticate themselves with the plug-in. If not present then the authentication method is considered to be by password entry (indicated by a value of **/Password**). The only other possible value that is currently defined is **/Fingerprint**. If other authentication types are required then these should be reported back to Adobe so they can be added to this specification. Intended to be used in claims of repudiation.
ADBE_PwdTime	number	(*Optional*) The number of seconds since the last time the signer was required to authenticate themselves using, for example, a keyed-in password. Intended to be used in claims of repudiation.

3 Encryption Dictionary – format adbe.pkcs7.s3

The encryption dictionary is defined in the *PDF Reference*, which is available on the Adobe Partner web site at http://partners.adobe.com/asn/developer/technotes.html. In the encryption dictionary the only required attributes are the **Filter** and **V** key. Other attributes are custom to the particular security handler. It is proposed that the SubFilter key be considered a required attribute, as explained in Signature and Encryption Handler Interoperability.

There is only one public-key format currently being considered for the encryption dictionary. This PKCS#7 Encryption Format uses PKCS#7 objects to wrap recipient lists and decryption information.

3.1 PKCS#7 Encryption Format

Table 3 describes the encryption dictionary for this format, with detailed descriptions appearing following this table.

Table 3 *Encryption Dictionary Attributes for PKCS#7 Format*

Key	*Type*	*Semantics*
Filter	name	(*Required in all encrypted documents*) Name of security handler.
SubFilter	name	(*Required, if it is desired that other registered security handlers be able to decrypt this document*) A name describing the format of data contained in the encryption dictionary. Always **adbe.pkcs7.s3** or **adbe.pkcs7.s4**. The name indicates an

		Adobe-originated format based on PKCS#7, with the encrypted-data contained in this data being used as described in Public Key Encryption. The **s3** and **s4** are for simple format version 3 and 4.
R (Revision)	number	(*Optional*) Specifies the version of the security handler that is being used. This value is specific to the security handler.
Recipients	array of strings	(*Required*) Contains an array of PKCS#7 objects where each PKCS#7 object lists recipients that have all been granted equal access rights to the PDF file encrypted data. There would normally be as many PKCS#7 objects in this array as there are unique sets of access permissions granted for the PDF file. The PKCS#7 objects in its enveloped data contain both the access permissions and the information used in the generation of the RC4 key that decrypts data in the file. The first PKCS#7 object should contain a list of all recipients that have been granted full *owner* access rights to data in the PDF file: this would include the originator of the file. A recipient should normally not appear in more then one PKCS#7 object. Access rights and permissions are as defined in table 6.46 of the PDF Reference Manual.

Public-Key Encryption

Public-key encryption allows multiple recipients to be specified for a document. A list of recipients with like access permissions is kept. Keys are encrypted separately for each recipient using his or her public key. Recipients can decrypt these keys using the recipient’s private key.

Only users in one of the PKCS#7 objects in the **Recipients** array will be able to open the document. This is different from the Standard security handler: the Standard security handler allows a situation where someone with an owner password has full access and, if there is no specified user password, the file can be opened by anyone without the need of a password.

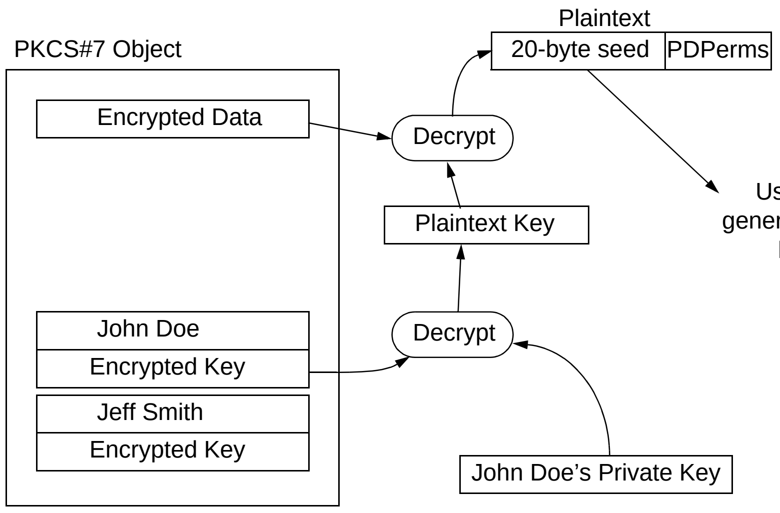
Encrypted data PKCS#7 Object

PKCS#7 is a Public Key Cryptographic Standard from the RSA Security web site at http://www.rsasecurity.com/. The PKCS#7 specification can be found at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/index.html.

PKCS#7 is also referred to as the *Cryptographic Message Syntax* (CMS). PKCS#7 is an object specification. This object is binary encoded and contains various attributes. When used to encrypt data a PKCS#7 object is designed to encapsulate a single encrypted copy of the data to be encrypted. The PKCS#7 object also contains a list of recipients for the data and, for each recipient, an encrypted key that can be used to decrypt the encrypted data. This encrypted key is encrypted using the recipients public key.

Public Key Encryption

The standard PDCryptHandler and RC4 encryption that has existed since Acrobat 2.1 are used. Only streams and strings are encrypted, and these are left in-place in the PDF file. This makes traditional PKCS#7 encapsulation of the encrypted data impossible. Instead the PKCS#7 object is used to encrypt and encapsulate the keying material that was used to generate the RC4 key used for PDF encryption. This results in an additional level of indirection, and three levels of encryption, before a user can access data in the file.



With the encryption dictionary format **adbe.pkcs7.s3** or **adbe.pkcs7.s4**, the encrypted data that is encapsulated into a PKCS#7 object by the security handler is a byte array that includes a 20-byte seed followed by 4-bytes that define the permissions (least significant byte of the permissions is placed first in the byte array). The encryption algorithm used to encrypt this encapsulated data is specified with whatever algorithms are available to the specific security handler. The seed value is used to create the crypt key so is a unique random number that the security handler will generate when creating the encrypted document.

With **adbe.pkcs7.s3** the 4-bytes of permissions are the same as is defined for the *Standard* security handler for revision **R** equal to 2. These permission bits are described in the *PDF Reference, Second Addition* in *Table 3.14 User password access privileges*. **adbe.pkcs7.s3** was used by the Entrust plug-in for Acrobat 4.x and 5.0.

With **adbe.pkcs7.s4** the 4-bytes of permissions are the same as is defined for the *Standard* security handler for revision **R** equal to 3. As of January 2001 an update was not available for the PDF Reference that included a description of revision 3. **adbe.pkcs7.s4** was used by the Acrobat Self-Sign plug-in for Acrobat 5.0.

The RC4 key that is used to encrypt the document is calculated using the seed obtained from the encrypted, encapsulated data in the PKCS#7 object as follows:

1. Initialize an SHA-1 message digest operation

2. Add the 20-bytes of seed to the SHA-1 message digest

3. Add the bytes of all values (PKCS#7 objects) in the array of **Recipients**, in the order in which they appear in the array, to the SHA-1 message digest.

4. Complete the SHA-1 message digest operation
5. Use the first (N/8)-bytes of the message digest as the RC4 key that can decrypt data in the PDF file, where N is the bit length of the RC4 key.

Note that the RC4 key is derived from a seed and a hash of the two PKCS#7 objects rather then being encrypted directly into the PKCS#7 object. This is done in order to make the encryption dictionary tamper proof.

4 Revision History

Revision	Date	Author	Description
3.2	2001.09.12	Jim Pravetz	Clarified seed value and fixed documentation error.
3.1	2001.02.20	Jim Pravetz	Clarified use of PKCS#7 format **adbe.pkcs7.sha1**.
3.0	2001.01.17	Jim Pravetz	Added **adbe.pkcs7.s4**.
2.02	1999.03.11	Jim Pravetz	Acrobat 4.0 final revision, posted to external web site.
0.1	1998.04	Jim Pravetz	Created