An Overview of the PKCS Standards

An RSA Laboratories Technical Note
Burton S. Kaliski Jr.
Revised November 1, 1993[*]

Abstract. *This note gives an overview of the PKCS family of standards for public-key cryptography. These standards cover RSA encryption, Diffie-Hellman key agreement, password-based encryption, extended-certificate syntax, cryptographic message syntax, private-key information syntax, and certification request syntax, as well as selected attributes. The note gives the motivation for the standards and discusses their relationship to other standards or agreements on public-key cryptography.*

1. Introduction

As public-key cryptography begins to see wide application and acceptance one thing is increasingly clear: If it is going to be as effective as the underlying technology allows it to be, there must be interoperable standards. Even though vendors may agree on the basic public-key techniques, compatibility between implementations is by no means guaranteed. Interoperability requires strict adherence to an agreed-upon standard format for transferred data. The standards described here provide such a basis for interoperability.

We call the standards described here "Public-Key Cryptography Standards," or "PKCS" for short. The standards consist of a number of components, called PKCS #1, #3, #5, #6, #7, #8, #9 and #10.[1]

The standards presented here evolved from the following broad design goals:

1. To maintain compatibility with PEM (the Internet Privacy-Enhanced Mail protocols, described in RFCs 1421–1424) wherever possible, at least to the extent of being able to share certificates and to translate encrypted

[*] Supersedes June 3, 1991 version, which was also published as NIST/OSI Implementors' Workshop document SEC-SIG-91-16. PKCS documents are available by electronic mail to <pkcs@rsa.com>.

[1] PKCS #2 and #4 are no longer active; both have been incorporated into the current PKCS #1.

and/or signed messages back and forth between PEM and PKCS.

2. To extend beyond PEM in being able to handle arbitrary binary data (not just ASCII data), to handle a richer set of attributes in (extended) certificates, to handle Diffie-Hellman key agreement [DH76], and to handle a richer set of features in digitally signed and enveloped data.
3. To describe a standard suitable for incorporation in future Open Systems Interconnection (OSI, described in X.200) standards. The standards here are based on the use of OSI standard ASN.1 (Abstract Syntax Notation One, described in X.208) and BER (Basic Encoding Rules, described in X.209) to describe and represent data.

PKCS describes the syntax for messages in an abstract manner, and gives complete details about algorithms. However, it does not specify how messages are to be represented, though BER is the logical choice. Thus PKCS implementations are free to exchange messages in any manner, depending on character set, record size constraints, and the like, as long as the abstract meaning of the messages can be preserved from sender to recipient.

The PKCS standards are offered by RSA Laboratories to developers of computer systems employing public-key technology. It is RSA Laboratories' intention to improve and refine the standards in conjunction with computer system developers, with the goal of producing standards that most if not all developers adopt.

The role of RSA Laboratories in the standards-making process is five-fold:

1. Publish carefully written documents describing the standards.
2. Retain sole decision-making authority on what each standard is. This includes arbitrary object identifier choices, etc.
3. Solicit opinions and advice from developers on useful or necessary changes and extensions.
4. Publish revised standards when appropriate.
5. Provide implementation guides and/or reference implementations.

Thus the standards-making process is not the usual committee-oriented method.

This note is divided into seven sections including this one. Section 2 gives some terminology. Section 3 addresses the question, "What needs to be standardized?" Section 4 summarizes the PKCS family and Section 5 compares PKCS with other standards. Section 6 presents some open issues and Section 7 concludes the note.

2. Background information

This section gives the basic background information necessary to understand the terminology in this note. The background information covers three areas: public-key cryptography, secret-key cryptography, and message-digest algorithms. For a more comprehensive background, the reader is referred to any of several nice survey articles [Riv90][Dif88][DH79].

2.1 Public-key cryptography

Public-key cryptography is the technology first identified by Diffie and Hellman [DH76] in which encryption and decryption involve different keys. The two keys are the *public key* and the *private key*, and either can encrypt or decrypt data. A user gives his or her public key to other users, keeping the private key to himself or herself. Data encrypted with a public key can be decrypted only with the corresponding private key, and vice versa.

A *public-key algorithm* is an algorithm for encrypting or decrypting data with a public or private key. A private key is typically used to encrypt a message digest (see Section 2.3); in such an application, the public-key algorithm is called a *message-digest encryption algorithm*. A public key is typically used to encrypt a content-encryption key (see Section 2.2); in such an application, the public-key algorithm is called a *key-encryption algorithm*.

A *signature algorithm* is an algorithm that transforms a message of any length under a private key to a *signature* in such a way that it is computationally infeasible to find two messages with the same signature, to find a message with a given, predetermined signature, or to find the signature of a given message without knowledge of the private key. Typically, a signature algorithm is implemented by computing a message digest on the message (see Section 2.3), then encrypting the message digest with the private key.

RSA is a public-key algorithm invented by Rivest, Shamir, and Adleman [RSA78] involving exponentiation modulo the product of two large prime numbers. The difficulty of breaking RSA is generally considered to be equal to the difficulty of factoring integers that are the product of two large prime numbers of approximately equal size.

Key agreement is a method whereby two parties, without prior arrangements, exchange messages in such a way that they agree upon a secret key that is known only to them. Key agreement can be achieved with a public-key algorithm, or with other methods. A *key-agreement algorithm* is an algorithm for achieving key agreement.

Diffie-Hellman is a key-agreement algorithm invented by Diffie and Hellman [DH76] involving exponentiation modulo a large prime number. The difficulty of breaking Diffie-Hellman is generally considered to be equal to the difficulty of computing discrete logarithms modulo a large prime number.

2.2 Secret-key cryptography

Secret-key cryptography is the technology in which encryption and decryption involve the same key, a *secret key*. Pairs of users share a secret key, keeping the key to themselves. Data encrypted with a secret key can be decrypted only with the same secret key.

A *secret-key algorithm* is an algorithm for encrypting or decrypting data with a secret key. A secret key is typically used to encrypt the content of a message; in such an application, the key is called a *content-encryption key* and the secret-key algorithm is called a *content-encryption algorithm*.

A *password-based encryption algorithm* is a secret-key algorithm in which the key is derived from a user-supplied password.

The Data Encryption Standard (DES) is the standard federal secret-key algorithm, described in FIPS PUB 46–1. *Cipher-Block Chaining (CBC)* is a mode of DES, described in FIPS PUB 81.

2.3 Message-digest algorithms

A *message-digest algorithm* is a method of reducing a message of any length to a string of a fixed length, called the *message digest*, in such a way that it is computationally infeasible to find a collision (two messages with the same message digest) or to find a message with a given, predetermined message digest.

MD2 and *MD5* are message-digest algorithms invented by RSA Laboratories, and are described in RFCs 1319 and 1321. Each inputs an arbitrary message and outputs a 128-bit message digest.

3. What needs to be standardized?

This section addresses the question, "What needs to be standardized?" To answer the question, we describe four applications of public-key cryptography: digital signature, digital enveloping, digital certification, and key agreement, looking at what aspects are suitable for standardization. Our emphasis is on those applications relevant to PKCS; there are certainly other applications, such as interactive authentication, that could be standardized.

The discussion of what needs to be standardized assumes two independent levels of abstraction. The first level is message syntax, and the second level is specific algorithms. The intention is that message syntax and specific algorithms should be orthogonal. For example, a standard for the syntax of digitally signed messages should be able to work with any public-key algorithm, not just RSA; and a standard for RSA should be applicable to many different message syntax standards.

The description of the four applications involves the usual cryptographic players Alice and Bob.

3.1 Digital signature

Digital signature is an application in which a signer, say "Alice," "signs" a message *m* in such a way that anyone can "verify" that the message was signed by no one other than Alice, and consequently that the message has not been modified since she signed it.

The typical implementation of digital signature involves a message-digest algorithm and a public-key algorithm for encrypting the message digest (i.e., a message-digest encryption algorithm):

- Alice reduces the message *m* to a message digest *d* with a message-digest algorithm; then she encrypts the message digest *d* with her private key, obtaining an encrypted message digest s. She sends the message *m* and the encrypted message digest s to Bob; the two parts together form the digitally signed message.
- Bob decrypts the encrypted message digest s with Alice's public key, obtaining the message digest *d*; then he reduces the message *m* to a comparative message digest *d'* and compares it to the message digest *d*. If the two are the same, he accepts the message.

Notice that Bob's work does not involve any information specific to him. Indeed, anyone can verify at any time that the message was signed by Alice, without access to any secret information. This application assumes that Bob knows Alice's public key; methods of developing trust in users' public keys are covered by the digital certificate application (Section 3.3).

Digital signature has three aspects that are suitable for standardization: an algorithm-independent syntax for digitally signed messages, specific message-digest algorithms, and specific public-key (message-digest encryption) algorithms.

Alice may also need a way to store her private key securely. One way to do this is to encrypt a message containing private-key information with a secret key derived from a password that Alice supplies. Aspects suitable for standardization here include an algorithm-independent syntax for encrypted private-key information, private-key syntax for specific public-key algorithms, and specific password-based encryption algorithms.

3.2 Digital enveloping

Digital enveloping is an application in which someone "seals" a message *m* in such a way that no one other than the intended recipient, say "Bob," can "open" the sealed message.

The typical implementation of digital enveloping involves a secret-key algorithm for encrypting the message (i.e., a content-encryption algorithm) and a public-key algorithm for encrypting the secret key (i.e., a key-encryption algorithm):

- Alice encrypts the message *m* with a randomly generated secret key *k*, obtaining an encrypted message *c*; then she encrypts the secret key *k* with Bob's public key, obtaining an encrypted secret key e. She sends the encrypted message *c* and the encrypted secret key e to Bob; the two parts together form the digitally enveloped message.

- Bob decrypts the encrypted secret key e with his private key, obtaining the secret key *k*; then he decrypts the encrypted message *c* with the secret key *k*, obtaining the message *m*.

Notice that Alice's work does not involve any information specific to her. Indeed, anyone can seal a message at any time for Bob, without access to any secret information. This application assumes that Alice knows Bob's public key; methods of developing trust in users' public keys are covered by the digital certificate application.

Digital enveloping has three aspects that are suitable for standardization: an algorithm-independent syntax for digitally enveloped messages, specific secret-key (content-encryption) algorithms, and specific public-key (key-encryption) algorithms.

Bob may need a way to store his private key securely, leading to similar aspects for standardization as those for digital signatures.

3.3 Digital certification

Digital certification is an application in which a certification authority "signs" a special message *m* containing the name of some user, say "Alice," and her public key in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in Alice's public key.

The typical implementation of digital certification involves a signature algorithm for signing the special message. (A signature algorithm is chosen here, rather than a message-digest algorithm followed by a message-digest encryption algorithm, as in the digital signature application, because X.509 certificates only use a signature algorithm.)

- Alice sends a "certification request" containing her name and her public key to a certification authority.

- The certification authority forms a special message *m* from Alice's request and signs the special message *m* under its private key, obtaining a signature s. The certification authority returns the message *m* and the signature s to Alice; the two parts together form a certificate.
- Alice sends the certificate to Bob to convey trust in her public key.
- Bob verifies the signature s under the certification authority's public key. If the signature verifies, he accepts Alice's public key.

As with an ordinary digital signature, anyone can verify at any time that the certificate was signed by the certification authority, without access to any secret information.

This application assumes that Bob knows the certification authority's public key. Bob can develop trust in the certification authority's public key recursively, if he has a certificate containing the certification authority's public key signed by a superior certification authority whom he already trusts. In this sense, a certificate is a stepping stone in digital trust. Ultimately, one need only trust the public keys of a small number of top-level certification authorities. Through a chain of certificates, trust in a large number of users' signatures can then be established.

A broader application of digital certification includes not only Alice's name and public key but also other information about Alice in the special message *m*. Such a message, together with a signature, forms what PKCS terms an extended certificate. Extended certificates are more than stepping stones in digital trust. They enable the certification authority not only to give Bob a means of trusting Alice's public key, but also that other information. The other information may include, for example, Alice's electronic-mail address, her authorization to sign documents of a given value, or her authorization to sign other certificates.

A certificate-revocation list (CRL) is another type of special message together with a signature. The special message for a CRL contains a list of revoked certificates, where the certificates are typically referenced indirectly by a serial number. A CRL enables the certification authority to "void" its signatures on Alice's certificate or extended certificates, as might be required when Alice's name changes or her private key is compromised.

Digital certification has six aspects that are suitable for standardization: an algorithm-independent syntax for certification requests, an algorithm-independent syntax for certificates, an algorithm-independent syntax for extended certificates, an algorithm-independent syntax for CRLs, public-key syntax for specific public-key algorithms, and specific signature algorithms.

3.4 Key agreement

Key agreement is an application in which Alice and Bob, without prior arrangements, exchange messages in such a way that they agree upon a secret key that is known only to them. The secret key can then be used, for example, to encrypt further communication between Alice and Bob.

The typical implementation of key agreement involves a two-phased key-agreement algorithm:

- Alice sends a message to Bob initiating the key-agreement protocol.
- Alice and Bob independently perform a first phase of some key-agreement algorithm, and send the result of that phase to one another.
- Alice and Bob independently perform a second phase of the key-agreement algorithm, after which they arrive at a common agreed-upon secret key.

Key agreement has two aspects that are suitable for standardization: an algorithm-independent syntax for key-agreement messages, and specific key-agreement algorithms.

3.5 Summary of useful standards

The foregoing discussion shows that following standards are useful in implementing digital signature, digital enveloping, digital certification, and key agreement:

1. Algorithm-independent syntax: digitally signed messages; digitally enveloped messages; certification requests; certificates; extended certificates; certificate-revocation lists; encrypted private-key information; key-agreement messages.
2. Algorithm-specific syntax: public keys; private keys.
3. Algorithms: message digest; secret-key encryption; public-key encryption; signature; password-based encryption; key agreement.

4. The PKCS standards

This section describes the members of the PKCS family. The descriptions of the members are largely taken from the PKCS documents themselves. Table 1 summarizes the correspondence between the PKCS standards and the syntax and algorithms suitable for standardization discussed in Section 3. When no PKCS is marked, the most applicable external works are listed.

PKCS leaves ample room for future expansion. Most objects defined by PKCS carry version numbers to allow backward compatibility in future revisions. Several of the objects also have space for arbitrary "attributes" that carry additional information not directly addressed by PKCS.

4.1 PKCS #1: RSA Encryption Standard

PKCS #1 describes a method, called rsaEncryption, for encrypting data using the RSA public-key cryptosystem. Its intended use is in the construction of digital signatures and digital envelopes, as described in PKCS #7:

- For digital signatures, the content to be signed is first reduced to a message digest with a message-digest algorithm (such as MD5), and then an octet string containing the message digest is encrypted with the RSA private key of the signer of the content. The content and the encrypted message digest are represented together according to the syntax in PKCS #7 to yield a digital signature. This application is compatible with Privacy-Enhanced Mail (PEM) methods.

Standard	**PKCS #**								**External work**
	1	**3**	**5**	**6**	**7**	**8**	**9**	**10**	
Algorithm-independent syntax:									
digitally signed messages					x		x		
digitally enveloped messages					x				
certification requests							x	x	
certificates									X.509, RFC 1422
extended certificates				x			x		
certificate-revocation lists									X.509, RFC 1422
encrypted private-key info.						x	x		
key agreement messages									[ISO90a], [ISO90b]
Algorithm-specific syntax:									
public keys: RSA	x								
private keys: RSA	x								

Algorithms:									
message digest: MD2, 5									RFCs 1319, 1321
secret-key encryption: DES									RFC 1423, [NIST92a]
public-key encryption: RSA	x								
signature: MD2, 4, 5 w/RSA	x								
password-based encryption			x						
key agreement: D-H		x							

Table 1. Correspondence between aspects suitable for standardization and PKCS.

- For digital envelopes, the content to be enveloped is first encrypted under a content-encryption key with a content-encryption algorithm (such as DES), and then the content-encryption key is encrypted with the RSA public key(s) of the recipient(s) of the content. The encrypted content and the encrypted content-encryption key are represented together according to the syntax in PKCS #7 to yield a digital envelope. This application is compatible with PEM methods.

PKCS #1 also describes a syntax for RSA public keys and private keys. The public-key syntax would be used in certificates; the private-key syntax would be used typically in encrypted private keys (PKCS #8). The public-key syntax is identical to that in both X.509 and PEM. Thus X.509/PEM RSA keys can be used in PKCS #1.

PKCS #1 also defines three signature algorithms, called md2WithRSAEncryption, md4WithRSAEncryption, and md5WithRSAEncryption, for use in signing X.509/PEM certificates and certificate-revocation lists, PKCS #6 extended certificates, and other objects employing digital signatures such as X.400 message tokens.

4.2 PKCS #3: Diffie-Hellman Key Agreement Standard

PKCS #3 describes a method for implementing Diffie-Hellman key agreement, whereby two parties, without any prior arrangements, can agree upon a secret key that is known only to them (and, in particular, is not known to an eavesdropper listening to the dialogue by which the parties agree on the key). This secret key can then be used, for example, to encrypt further communications between the parties.

The intended application of PKCS #3 is in protocols for establishing secure connections, such as those proposed for OSI's transport and the network layers [ISO90a][ISO90b].

4.3 PKCS #5: Password-Based Encryption Standard

PKCS #5 describes a method for encrypting an octet string with a secret key derived from a password. The result of the method is an octet string. Although PKCS #5 can be used to encrypt arbitrary octet strings, its intended primary application to public-key cryptography is for encrypting private keys when transferring them from one computer system to another, as described in PKCS #8.

PKCS #5 defines two key-encryption algorithms: pbeWithMD2AndDES-CBC and pbeWithMD5AndDES-CBC. The algorithms employ DES secret-key encryption in cipher-block chaining mode, where the secret key is derived from a password with the MD2 or MD5 message-digest algorithm.

4.4 PKCS #6: Extended-Certificate Syntax Standard

PKCS #6 describes a syntax for extended certificates. An extended certificate consists of an X.509 public-key certificate and a set of attributes, collectively signed by the issuer of the X.509 public-key certificate. Thus the attributes and the enclosed X.509 public-key certificate can be verified with a single public-key operation, and an ordinary X.509 certificate can be extracted if needed, e.g., for Privacy-Enhanced Mail.

The intention of including a set of attributes is to extend the certification process beyond just the public key to include other information about a given entity, such as electronic-mail address. A non-exhaustive list of attributes is given in PKCS #9.

The preliminary intended application of PKCS #6 is in the cryptographic-enhancement syntax standard (PKCS #7), but it is expected that other applications will be developed.

4.5 PKCS #7: Cryptographic Message Syntax Standard

PKCS #7 describes a general syntax for data that may have cryptography applied to it, such as digital signatures and digital envelopes. The syntax admits recursion, so that, for example, one envelope can be nested inside another, or one party can sign some previously enveloped digital data. It also allows arbitrary attributes, such as signing time, to be authenticated along with the content of a message, and provides for other attributes such as countersignatures to be associated with a signature. A degenerate case of the syntax provides a means for disseminating certificates and certificate-revocation lists.

PKCS #7 is compatible with Privacy-Enhanced Mail (PEM) in that signed-data and signed-and-enveloped-data content, constructed in a PEM-compatible mode, can be converted into PEM messages without any cryptographic operations. PEM messages can similarly be converted into the signed-data and signed-and-enveloped data content types.

PKCS #7 can support a variety of architectures for certificate-based key management, such as the one described for Privacy-Enhanced Mail in RFC 1422. Architectural decisions such as what certificate issuers are considered "top-level," what entities certificate issuers are authorized to certify, what distinguished names are considered acceptable, and what policies certificate issuers must follow (such as signing with secure hardware, or requiring entities to present specific forms of identification) are left outside PKCS #7. Dissemination of "hot lists" of invalid certificates (certificate-revocation lists) is also left outside.

The values produced according to PKCS #7 are intended to be BER-encoded, which means that the values would typically be represented as octet strings. While many systems are capable of transmitting arbitrary octet strings reliably, it is well known that many electronic-mail systems are not. PKCS #7 does not address mechanisms for encoding octet strings as (say) strings of ASCII characters or other techniques for enabling reliable transmission by re-encoding the octet string. RFC 1421 suggests one possible solution to this problem.

4.6 PKCS #8: Private-Key Information Syntax Standard

PKCS #8 describes a syntax for private-key information. Private-key information includes a private key for some public-key algorithm and a set of attributes. PKCS #8 also describes a syntax for encrypted private keys. A password-based encryption algorithm (e.g., one of those described in PKCS #5) could be used to encrypt the private-key information.

The intention of including a set of attributes is to provide a simple way for a user to establish trust in information such as a distinguished name or a top-level certification authority's public key. While such trust could also be established with a digital signature, encryption with a secret key known only to the user is just as effective and possibly easier to implement. A non-exhaustive list of attributes is given in PKCS #9.

4.7 PKCS #9: Selected Attribute Types

PKCS #9 defines selected attribute types for use in PKCS #6 extended certificates, PKCS #7 digitally signed messages, and PKCS #8 private-key information.

4.8 PKCS #10: Certification Request Syntax Standard

PKCS #10 describes a syntax for certification requests. A certification request consists of a distinguished name, a public key, and optionally a set of attributes, collectively signed by the entity requesting certification. Certification requests are sent to a certification authority, who transforms the request to an X.509 public-key certificate, or a PKCS #6 extended certificate. (In what form the certification authority returns the newly signed certificate is outside the scope of PKCS #10. A PKCS #7 message is one possibility.)

The intention of including a set of attributes is twofold: to provide other information about a given entity, such as the postal address to which the signed certificate should be returned if electronic mail is not available, or a "challenge password" by which the entity may later request certificate revocation; and to provide attributes for a PKCS #6 extended certificate. A non-exhaustive list of attributes is given in PKCS #9.

Certification authorities may also require non-electronic forms of request and may return non-electronic replies. It is expected that descriptions of such forms, which are outside the scope of PKCS #10, will be available from the certification authority.

The preliminary intended application of PKCS #10 is to support PKCS #7 cryptographic messages, but is expected that other applications will be developed.

5. Compatibility with other work

This section describes the compatibility of the PKCS standards with other standards or agreements on public-key cryptography. For simplicity, we refer to the various works involving public-key cryptography as "standards," without regard to their formal approval by a standards-making body.

Compatibility has many meanings. For instance, a standard A may be considered compatible with another standard B if standard A provides algorithms that standard B can use. Or, standard A may generate data that standard B can process directly. We choose the definition that standard A is compatible with standard B if standard A provides something useful to standard B, where the usefulness may be contingent on a change in representation, and possibly on omission of information. Cryptographic operations are not allowed in the change of representation.

We say standard A is "outbound" compatible with standard B if implementations of standard A produce something useful to implementations of standard B, but not necessarily vice versa, and we say standard A is "inbound" compatible with standard B if implementations of standard B produce something useful to implementations of standard A, but not necessarily vice versa.

We address compatibility with seven related works:

1. Privacy-Enhanced Mail, as defined in RFCs 1421–1424.
2. Directory Services—Authentication Framework, as defined in CCITT Recommendation X.509.
3. Message Handling Systems, as defined in CCITT Recommendation X.400.
4. Draft network-layer and transport-layer security protocols [ISO90a] [ISO90b].
5. NIST's proposed Digital Signature Standard and Secure Hash Standard, as defined in [NIST92] and FIPS PUB 180.
6. ISO/IEC 9796: Digital Signature Scheme Giving Message Recovery.
7. ANSI X9.30 and .31 (draft): Public-key cryptography with irreversible and reversible algorithms.

5.1 Privacy-Enhanced Mail

PKCS is inbound compatible with Privacy-Enhanced Mail, as defined in RFCs 1421–1424. With suitable restrictions, PKCS is outbound compatible as well.

5.1.1 Primary compatibilities

A privacy-enhanced message generated according the Privacy-Enhanced Mail RFCs can be converted to a form that can be processed by implementations of PKCS #7 without any cryptographic operations. The conversion process is "flat" in the sense that the encapsulated text of the privacy-enhanced message becomes the "inner" content of the PKCS #7 data. If the encapsulated text happens to contain privacy-enhanced messages, those messages are not interpreted in the conversion process.

Data with certain PKCS #7 cryptographic enhancements can be converted to a form that can be processed by implementations of the Privacy-Enhanced Mail RFCs.

Privacy-Enhanced Mail can effectively be viewed as a set of encoding rules, analogous to the Basic Encoding Rules for ASN.1, for PKCS #7 data with these restrictions. Conversion from PKCS #7 to PEM may involve omission of attributes from PKCS #6 extended certificates, which is acceptable since the attributes are not essential to PEM.

5.1.2 Further compatibilities

RSA encryption in PKCS #1, in block types 01 and 02, is the same as in PEM, as defined RFC 1423.

Certificates in PEM are one of the alternatives of PKCS #7's ExtendedCertificateOrCertificate type. (See the next section for more details.) The md2WithRSAEncryption and md5WithRSAEncryption signature algorithms in PKCS #1 are the same as PEM's message and certificate signature algorithms.

Certificate revocation lists (CRLs) in PEM are in PKCS #7's CertificateRevocationLists type.

5.2 Directory Services—Authentication Framework (X.509)

PKCS is compatible with Directory Services—Authentication Framework, as defined in CCITT Recommendation X.509.

5.2.1 Primary compatibilities

A certificate generated according to X.509 can be converted to a form that can be used in implementations of PKCS #7. The conversion involves the type ExtendedCertificateOrCertificate, which has two alternatives, an X.509 certificate and a PKCS #6 extended certificate.

An extended certificate generated according to PKCS #6 can be converted to a form that can be used in implementations of X.509, since an extended certificate contains an X.509 certificate. The conversion involves the omission of extended attributes.

5.2.2 Further compatibilities

RSA private-key encryption in PKCS #1 is the same, in block type 00, as RSA private-key encryption in X.509.

The signature process for X.509 certificates is the same as the signature process for PKCS #6 extended certificates. That is, both use X.509's SIGNED macro (or an equivalent form), so both can use any signature algorithm consistent with the SIGNED macro.

The md2WithRSAEncryption and md5WithRSAEncryption signature algorithms in PKCS #1 are consistent with the SIGNED macro, in that they input an octet string and output a bit string. Thus, they can be used in signing X.509 certificates, or any other quantity signed in the authentication framework or in other uses of the SIGNED macro (e.g., in X.411 security—see Section 5.3.2).

RSA public-key syntax in X.509 Annex C is the same as RSA public-key syntax in PKCS #1.

5.2.3 Incompatibilities

RSA encryption in PKCS #1 is different than RSA encryption in X.509, in that the latter does not specify any method of padding the quantity input to encryption.

The rsaEncryption algorithm is inconsistent with X.509's SIGNED and ENCRYPTED macros, in that it outputs an octet string, not a bit string.

The pbeWithMD2AndDES-CBC and pbeWithMD5AndDES-CBC password-based encryption algorithms in PKCS #6 are inconsistent with X.509's ENCRYPTED macro, in that they output an octet string, not a bit string. (However, it is not difficult to convert from one form to another.)

A certificate revocation list (CRL) generated according to X.509 is not compatible with Privacy-Enhanced Mail, as defined in RFCs 1421–1424, and hence is not compatible with PKCS #7. (Corrections to X.509 RFCs are being considered.)

The syntax for encrypted private-key information in PKCS #8 does not use X.509's ENCRYPTED macro, or an equivalent form. (The encrypted private key is represented as an octet string, not as a bit string, as the ENCRYPTED macro assumes.) Thus, encryption algorithms consistent with X.509's ENCRYPTED macro are not useful in PKCS #8.

5.3 Message Handling Systems (X.400)

PKCS is outbound compatible with Message Handling Systems, as defined in CCITT Recommendation X.400, under suitable restrictions, and with the appropriate unauthenticated attributes. (This does not mean that PKCS provides sufficient information to build an X.400 message, just that X.400-compatible cryptographic enhancements can be computed.) PKCS is not inbound compatible with X.400.

5.3.1 Primary compatibilities

Data with certain PKCS #7 cryptographic enhancements and appropriate unauthenticated attributes can be converted into a form that can be processed by implementations of the X.400 security services. The restrictions on the cryptographic enhancements include the following:

- the "outer" content type must be signedData
- the "inner" content type must be data

The reason that the "outer" content type must be signedData is that the "inner" content must be presented in the clear, since encrypted content in PKCS #7 is different than encrypted content in X.400. The latter encrypts a complete BER encoding, and the former encrypts only the contents octets.

Compatibility with X.400 is achieved by placing an X.411 message token among the unauthenticated attributes for the signer of the PKCS #7 data. Computing the X.411 message token involves another private-key operation with the signer's private key in addition to the one for computing the signer's encrypted message digest already required by PKCS #7, so X.400 compatibility is not efficient.

5.3.2 Further compatibilities

Since the md2WithRSAEncryption and md5WithRSAEncryption signature algorithms in PKCS #1 are consistent with the SIGNED and SIGNATURE macros, as discussed in Section 5.2.2, those algorithms can be used in computing these X.411 quantities: content-integrity check; message origin-authentication check; and asymmetric token.

5.4 Draft network-layer and transport-layer security protocols

PKCS is compatible with the draft standards for security in the network and transport layer [ISO90a][ISO90b]. Specifically, the dhKeyAgreement algorithm in PKCS #3 can be used in either of those draft standards.

5.5 DSS and SHS

PKCS is partially compatible with NIST's proposed Digital Signature Standard (DSS). PKCS #6 extended certificates may be signed with DSS, but since DSS signatures do not include a PKCS #7 DigestInfo value, they are not compatible with PKCS #7.

PKCS is compatible with the Secure Hash Standard (SHS), which can be used as a message-digest algorithm in PKCS #7.

5.6 ISO/IEC 9796

PKCS is only partially compatible with the ISO/IEC standard digital signature scheme giving message recovery. PKCS #6 extended certificates and PKCS #7 signed-data content may be signed according to ISO/IEC 9796. However, PKCS #1 is not compatible, as the RSA encryption block format in PKCS #1 is different than the format specified by ISO/IEC 9796.

5.7 ANSI X9.30 and .31

PKCS is partially compatible with the draft X9.30 and .31 for public-key cryptography with irreversible and reversible algorithms. Specifically, signatures in X9.31-1 are based on DSS and those in X9.30-1 are based on ISO/IEC 9796, each of which is partially compatible with PKCS. It remains to be seen whether X9.30 and .31's key management will be compatible with PKCS digital envelopes.

Certification requests in the draft X9.31-3 are similar to those in the new PKCS #10, but not compatible. (Later versions may well be compatible.)

6. Open issues

While PKCS provides a basis for interoperability between implementations of public-key cryptography, some issues relevant to the meaningful interaction of implementations remain open. Two implementations of PKCS may be able to complete the four applications in Section 3 successfully, but may have difficulty agreeing on the meaning of that success without further agreement on certain issues: names and the certification hierarchy. Furthermore, some issues are explicitly left outside of the scope of PKCS, such as security conditions on the choice of key.

This section summarizes the open issues in naming, the certification hierarchy, and security conditions.

6.1 Naming

Naming of entities is a complicated issue. In adopting X.509 certificates for compatibility with PEM, PKCS also adopts X.500 distinguished names, and inherits their complexity. Basically, an X.500 distinguished name defines a "path" through an X.500 directory tree from the root of the tree to an object of interest. Given that PKCS, like PEM, is being developed in advance of widespread deployment of X.500 directories, it is not clear what most objects' (i.e., Alice's or Bob's) distinguished names are. Some effort is underway to establish conventions for naming (see RFC 1255, X.521, or RFC 1422), and implementors of PKCS should anticipate these conventions when constructing names. However, there is no guarantee that an entity's name chosen today will be the same as the one assigned by an X.500 directory administrator in the future. Consequently, certificates constructed today may not necessarily be meaningful to X.500 implementations in the future.

(An example of an X.500 directory name is presented in the guide to ASN.1 and BER [Kal93].)

Some of the open issues in naming include:

- maximum length of the name in terms of number of arcs (relative distinguished names) in the path
- constraints on the relative distinguished names (specifically, the maximum number of "attribute-value assertions" in an arc, the allowed set of attributes, and upper limits on the lengths of values)
- conventions for names of particular types of object, e.g., organizations, residential persons, organizational persons, etc.
- character-set concerns, such as which extensions to the T.61 character set are accepted, and when to choose, for example, T.61 as opposed to the 16-bit Universal Character Set

RSA Laboratories intends to monitor conventions for naming and to report any progress in appendices to future releases of PKCS.

6.2 Certification

Another complicated issue is the meaning of certification: specifically, who is trusted to issue certificates, and to whom. Syntactically, any entity can sign a certificate as issuer with any entity as subject. Practically speaking, one would like to have some manner of filtering out certificates whose issuer-to-subject relationship is questionable. For instance, one would probably question a certificate issued by one company to employees of another company. One would also like to bound the length of certificate chains so that the chains can be found and represented easily. As with names, some work is underway to establish conventions for certification (see RFC 1422).

Open issues here include:

- what level of trust in the subject's identity is implied by a certificate
- the correspondence between the directory tree and issuer-to-subject relationships
- which entities can act as top-level certification authorities, having their public keys widely known
- the maximum length of a certificate chain

Some of the certification issues can be resolved with PKCS #6 extended certificates. For instance, one could define an extended-certificate attribute that indicates the authority of a certificate's subject to issue other certificates. Another attribute could indicate to what extent the subject can delegate authority. Such techniques are employed in the Electronic Document Authorization architecture [Fis90], but would require further study before being included in PKCS.

Again, RSA Laboratories intends to monitor conventions for certification, and to report any progress in appendices to future releases of PKCS.

6.3 Security conditions

The three algorithm standards—PKCS #1 (RSA Encryption Standard), PKCS #3 (Diffie-Hellman Key Agreement standard), and PKCS #5 (Password-Based Encryption Standard)—all involve security conditions on the choice of key (or password, in the case of PKCS #5). Such conditions may change as the state of the art in cryptanalysis improves, and are subject to tradeoffs between performance and security. For example, the conventional argument that the factors of the RSA modulus should be strong primes seems no longer to hold [Riv91], which is why PKCS neither mandates strong primes, nor discourages their use. Since security conditions do not affect the format of transferred data, the security conditions are left outside the scope of PKCS.

Specific open issues, left to implementors, include:

- range of lengths of RSA modulus n in PKCS #1 (for example, RFC 1423 sets the range as 508 to 1024 bits);
- conditions on RSA primes p and q, such as whether p-1 and q-1 should

have large factors, and how far apart *p* and *q* should be;

- additional conditions on the RSA public exponent *e* and the RSA private exponent *d*;
- range of lengths of the Diffie-Hellman modulus *p* in PKCS #3;
- conditions on the Diffie-Hellman modulus *p*, such as whether *p*-1 should have a large factor;
- conditions on the Diffie-Hellman base *g*, such as how large a group it should generate (e.g., all nonzero elements modulo *p*);
- length of the Diffie-Hellman private value *x*;
- range of lengths of the password *P* in PKCS #5;
- structural requirements on the password *P* (e.g., at least one non-alphanumeric character); and
- sources of pseudorandom bits in all the algorithm standards.

It is RSA Laboratories' intention to release "recommended practices" documents from time to time that address security conditions such as those just listed.

7. Conclusion

The PKCS family of standards addresses the following need: an agreed-upon standard format for transferred data based on public-key cryptography. PKCS covers several aspects of public-key cryptography, including RSA encryption, Diffie-Hellman key agreement, password-based encryption, extended-certificate syntax, cryptographic-enhancement syntax, and private-key information syntax. PKCS evolved from three broad design goals: to maintain compatibility with Privacy-Enhanced Mail, to extend beyond PEM, and to be suitable for incorporation in future OSI standards.

This note has summarized PKCS. It has shown that PKCS provides a basis for interoperability in the several areas of interest, and that PKCS has a high level of PEM compatibility, several extensions, and significant compatibility with existing OSI standards. The note has also identified some open issues outside the scope of PKCS. The reader is encouraged to review and implement PKCS and to make constructive comments.

References

FIPS PUB 46–1 National Bureau of Standards. *FIPS PUB 46–1: Data Encryption Standard.* January 1988.

FIPS PUB 81 National Bureau of Standards. *FIPS PUB 81: DES Modes of Operation.* December 1980.

FIPS PUB 180 National Institute of Standards and Technology. *FIPS PUB 180: Secure Hash Standard (SHS).* May 11, 1993.

ISO/IEC 9796 ISO/IEC. *ISO/IEC 9796: Digital signature scheme giving message recovery*. October 1991.

PKCS #1 RSA Laboratories. *PKCS #1: RSA Encryption Standard.* Version 1.5, November 1993.

PKCS #3 RSA Laboratories. *PKCS #3: Diffie-Hellman Key-Agreement Standard.* Version 1.4, November 1993.

PKCS #5 RSA Laboratories. *PKCS #5: Password-Based Encryption Standard.* Version 1.5, November 1993.

PKCS #6 RSA Laboratories. *PKCS #6: Extended-Certificate Syntax Standard.* Version 1.5, November 1993.

PKCS #7 RSA Laboratories. *PKCS #7: Cryptographic Message Syntax Standard.* Version 1.5, November 1993.

PKCS #8 RSA Laboratories. *PKCS #8: Private-Key Information Syntax Standard.* Version 1.2, November 1993.

PKCS #9 RSA Laboratories. *PKCS #9: Selected Attribute Types.* Version 1.1, November 1993.

PKCS #10 RSA Laboratories. *PKCS #10: Certification Request Syntax Standard.* Version 1.0, November 1993.

RFC 1255 The North American Directory Forum. *RFC 1255: A Naming Scheme for c=US.* September 1991. (Also published as *NADF-175: A Naming Scheme for c=US.* July 1991.)

RFC 1319 B. Kaliski. *RFC 1319: The MD2 Message-Digest Algorithm.* April 1992.

RFC 1321 R. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm.* April 1992.

RFC 1421 J. Linn. *RFC 1421: Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures.* February 1993.

RFC 1422 S. Kent. *RFC 1422: Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management.* February 1993.

RFC 1423 D. Balenson. *RFC 1423: Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers.* February 1993.

RFC 1424 B. Kaliski. *RFC 1424: Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services.* February 1993.

X.200 CCITT. *Recommendation X.200: Reference Model of Open Systems Interconnection for CCITT Applications.* 1984.

X.208 CCITT. *Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1).* 1988.

X.209 CCITT. *Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).* 1988.

X.400 CCITT. *Recommendation X.400: Message Handling System and Service Overview.*

1988.

X.411 CCITT. *Recommendation X.411: Message Handling Systems: Message Transfer System: Abstract Service Definition and Procedures.* 1988.

X.500 CCITT. *Recommendation X.500: The Directory—Overview of Concepts, Models and Services.* 1988.

X.509 CCITT. *Recommendation X.509: The Directory—Authentication Framework.* 1988.

X.521 CCITT. *Recommendation X.521: The Directory—Selected Object Classes.* 1988.

X9.30-1 Accredited Standards Committee X9. *American National Standard X9.30-199X: Public Key Cryptography using Irreversible Algorithms for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA).* Draft, June 18, 1993.

X9.30-3 Accredited Standards Committee X9. *American National Standard X9.30-199X: Public Key Cryptography using Irreversible Algorithms for the Financial Services Industry: Part 3: Certificate Management for DSA.* Draft, September 27, 1993.

X9.31-1 Accredited Standards Committee X9. *American National Standard X9.31-1992: Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry: Part 1: The RSA Signature Algorithm.* Draft, March 7, 1993.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory,* IT-22:644–654, 1976.

[DH79] W. Diffie and M.E. Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE,* 67(3):397–427, March 1979.

[Dif88] W. Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE,* 76(5):560–577, May 1988.

[Fis90] A. Fischer. Electronic document authorization. In *Proceedings of the 13th National Computer Security Conference.* 1990.

[ISO90a] ISO. *JTC1/SC6/N6285: Draft Transport Layer Security Protocol.* Draft, November 1990.

[ISO90b] ISO. *JTC1/SC6/N2559: Draft Network Layer Security Protocol.* Draft, September 1990.

[Kal93] Burton S. Kaliski Jr. *A Layman's Guide to a Subset of ASN.1, BER, and DER.* RSA Laboratories, November 1993.

[NIST92] National Institute of Standards and Technology. *Publication XX: Announcement and Specifications for a Digital Signature Standard (DSS).* August 19, 1992.

[NIST92a] National Institute for Standards and Technology. *Special Publication 500-202: Stable Implementation Agreements for Open Systems Interconnection Protocols*. Part 12 (Security). December 1992.

[Riv90] Ronald L. Rivest. Cryptography. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1, pages 719–755. Elsevier Science, 1990.

[Riv91] Ronald L. Rivest. Are "strong" primes needed for RSA? Unpublished manuscript, May

1991.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

Revision history

June 3, 1991 version

The June 3, 1991 version is part of the initial public release of PKCS. It was published as NIST/OSI Implementors' Workshop document SEC-SIG-91-16.

November 1, 1993 version

The November 1, 1993 version incorporates several editorial changes, including the addition of a revision history. It is updated to be consistent with the following versions of the PKCS documents:

PKCS #1: RSA Encryption Standard. Version 1.5, November 1993.
PKCS #3: Diffie-Hellman Key-Agreement Standard. Version 1.4, November 1993.
PKCS #5: Password-Based Encryption Standard. Version 1.5, November 1993.
PKCS #6: Extended-Certificate Syntax Standard. Version 1.5, November 1993.
PKCS #7: Cryptographic Message Syntax Standard. Version 1.5, November 1993.
PKCS #8: Private-Key Information Syntax Standard. Version 1.2, November 1993.
PKCS #9: Selected Attribute Types. Version 1.1, November 1993.
PKCS #10: Certification Request Syntax Standard. Version 1.0, November 1993.

The following substantive changes were made:

Section 5: Compatibility with NIST's proposed Digital Signature Standard and Secure Hash Standard, ISO/IEC 9796, and ANSI X9.30 and .31 is discussed.

Author's address

Burton S. Kaliski Jr., Ph.D.
Chief Scientist
RSA Laboratories
100 Marine Parkway
Redwood City, CA 94065 USA

(415) 595-7703
(415) 595-4126 (fax)
burt@rsa.com