PKCS #3:
Diffie-Hellman Key-Agreement Standard

RSA Data Security, Inc.
Redwood City, CA

June 3, 1991

Version 1.3[1]

1. Scope

This standard describes a method for implementing Diffie-Hellman key agreement [1], 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 this standard is in protocols for establishing secure connections, such as those proposed for the transport layer [2] and the network layer [3].

Details on the interpretation of the agreed-upon secret key are outside the scope of this standard, as are details on sources of the pseudorandom bits required by this standard.

2. References

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

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

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

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

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

---

[1] Versions 1.0–1.2 are superseded by this version. The earlier versions were distributed to participants in RSA Data Security, Inc.'s Public-Key Cryptography Standards meetings in February and March 1991.

3. Definitions

For the purposes of this standard, the following definitions apply.

`AlgorithmIdentifier`: A type that identifies an algorithm (by object identifier) and any associated parameters, as defined in X.509 [4].

ASN.1: Abstract Syntax Notation One, as defined in X.208 [5].

Diffie-Hellman key: Prime and base.

4. Symbols and abbreviations

Upper-case italic symbols (e.g., *PV*) denote octet strings; lower-case italic symbols (e.g., *i*) denote integers.

PV	public value
PV′	other entity's public value
SK	secret key
g	base
k	length of prime in octets
p	prime
x	private value
x′	other entity's private value
y	integer public value
y′	other entity's integer public value
z	integer secret key
mod *n*	arithmetic computation modulo *n*
‖*X*‖	length in octets of octet string *X*

5. General overview

The next four sections specify key generation, two phases of Diffie-Hellman key agreement, and an object identifier.

A central authority shall generate a Diffie-Hellman key, and the two phases of key agreement shall be performed with this key. It is possible that more than one key may be generated by a given central authority, and that there may be more than one central authority. The algorithm identifier for Diffie-Hellman key agreement specifies which Diffie-Hellman key is employed.

Two entities intending to agree on a secret key shall employ the first phase independently to produce outputs *PV* and *PV'*, the public values. The entities shall exchange the outputs.

The entities shall then employ the second phase independently with the other entity's public value as input. The mathematics of Diffie-Hellman key agreement ensure that the outputs *SK* of the second phase are the same for both entities.

6. Key generation

This section describes Diffie-Hellman key generation.

A central authority shall select an odd prime p. The central authority shall also select an integer g, the base, that satisfies $0 < g < p$.

The length of the prime p in octets is the integer k satisfying

$$2^{8(k-1)} \leq p < 2^{8k}$$

Note. Some additional conditions on the choice of prime and base may well be taken into account in order to deter discrete logarithm computation. These security conditions fall outside the scope of this standard.

7. Phase I

This section describes the first phase of Diffie-Hellman key agreement.

The first phase consists of three steps: private-value generation, exponentiation, and integer-to-octet-string conversion. The input to the first phase shall be the Diffie-Hellman key (p and g). The output from the first phase shall be an octet string *PV*, the public value; and an integer x, the private value.

This phase is performed independently by the two parties intending to agree on a secret key.

7.1 Private-value generation

An integer x, the private value, shall be generated privately and randomly. This integer shall satisfy $0 < x < p-1$.

7.2 Exponentiation

The base g shall be raised to the private value x modulo p to give an integer y, the integer public value.

$$y = g^x \bmod p, \quad 0 < y < p$$

This is the classic discrete-exponentiation computation.

7.3 Integer-to-octet-string conversion

The integer public value y shall be converted to an octet string PV of length k, the public value. The public value PV shall satisfy

$$y = \sum_{i=1}^{k} 2^{8(k-i)} PV_i \tag{1}$$

where $PV_1, \ldots, PV_k$ are the octets of PV from first to last.

In other words, the first octet of PV has the most significance in the integer and the last octet of PV has the least significance.

8. Phase II

This section describes the second phase of Diffie-Hellman key agreement.

The second phase consists of three steps: octet-string-to-integer conversion, exponentiation, and integer-to-octet-string conversion. The input to the second phase shall be the Diffie-Hellman key (only p); an octet string PV', the other entity's public value; and the private value x. The output from the second phase shall be an octet string SK, the agreed-upon secret key.

This phase is performed independently by the two parties intending to agree on a secret key, after the parties have exchanged public values resulting from the first phase.

8.1 Octet-string-to-integer conversion

The other entity's public value PV' shall be converted to an integer y', the other entity's integer public value. Let $PV'_1, \ldots, PV'_k$ be the octets of PV' from first to last. Then the other entity's integer public value y' shall satisfy

$$y' = \sum_{i=1}^{k} 2^{8(k-i)} PV'_i$$

In other words, the first octet of PV' has the most significance in the integer and the last octet of PV' has the least significance.

8.2 Exponentiation

The other entity's integer public value y' shall be raised to the private integer x modulo p to give an integer z, the integer secret key.

$$z = (y')^x \bmod p, \quad 0 < z < p$$

This is the classic discrete-exponentiation computation.

Note. The integer secret key z satisfies

$$z = (y')^x = (g^{x'})^x = (g^x)^{x'} = y^{x'} \bmod p$$

where x' is the other entity's private value. This mathematical relationship is the reason the two entities arrive at the same key.

8.3 Integer-to-octet-string conversion

The integer secret key z shall be converted to an octet string SK, the secret key, of length k. The secret key SK shall satisfy

$$z = \sum_{i=1}^{k} 2^{8(k-i)} SK_i$$

where $SK_1, \ldots, SK_k$ are the octets of SK from first to last.

In other words, the first octet of *SK* has the most significance in the integer and the last octet of *SK* has the least significance.

9. Object identifier

This standard defines two object identifiers: `pkcs-3` and `dhKeyAgreement`.

9.1 pkcs-3

The object identifier `pkcs-3` identifies this standard.

```
pkcs-3 OBJECT IDENTIFIER ::=
  { iso(1) member-body(2) US(840) rsadsi(113549) pkcs(1) 3 }
```

The `pkcs-3` object identifier is intended to be used only in defining other object identifiers in this standard.

9.2 dhKeyAgreement

The object identifier `dhKeyAgreement` identifies the Diffie-Hellman key agreement method defined in defined in Sections 7 and 8.

```
dhKeyAgreement OBJECT IDENTIFIER ::= { pkcs-3 1 }
```

The `dhKeyAgreement` object identifier is intended to be used in the `algorithm` field of a value of type `AlgorithmIdentifier`. The `parameters` field of that type, which has the algorithm-specific syntax `ANY DEFINED BY algorithm`, would have ASN.1 type `DHParameter` for this algorithm.

```
DHParameter ::= SEQUENCE {
  prime INTEGER, -- p
  base INTEGER -- g }
```

The fields of type `DHParameter` have the following meanings:

- `prime` is the prime *p*.
- `base` is the base *g*.