LUC Public-Key Cryptosystem & Digital Signature

9.1 ISO Entry Name

{iso standard 9979 luc-pkcds (3)}

9.2 Proprietary Entry Name

LUC Public-Key Cryptosystem and Digital Signature

9.3 Intended Range of Applications

a) confidentiality
b) authentication
c) data integrity
d) digital signatures
e) digital envelopes
f) Electronic Data Interchange (EDI)

9.4 Cryptographic Interface Parameters

These are discussed in the accompanying document "LUC Encryption Standard".

9.5 Test Words

Two examples of encryption/decryption are given. It should be noted that making and verifying a LUC digital signature is the same as below, except that the decryption stage is done before the encryption.

Let N = pq = 1949x2089 = 4071461 and let P = 11111 be the message to encrypt and decrypt. The public keys will be e and N; the particular private key for 11111 has four possible values: d1, d2, d3, & d4.

First, calculate the Lehmer totient function of P with respect to N. To do this we need to calculate the Legendre function (as explained in books on elementary number theory) of p and q with respect to D (= P^2 - 4). This also takes four values, depending on the message.

Let D = P^2 - 4; then (D/1949) = -1 and (D/2089) = -1 are the two Legendre values. Hence the Lehmer totient is the Least Common Multiple of (1949 + 1) and (2089 + 1).

(D/1949)	(D/2089)	**Lehmer totient value(r)**
+	+1	LCM(1949-1, 2089-1)

+1	-1	LCM(1949-1, 2089+1)
-1	+1	LCM(1949+1, 2089-1)
-1	-1	LCM(1949+1, 2089+1)

Choosing e = 1103 for our public key, we use the Extended Euclidean Algorithm (see Knuth's book) to find the secret key d, to solve the modular equation $ed \equiv 1$ modulo r. d turns out to equal 24017. To encrypt the message 11111, we make the calculation:

$$V_{1103}(11111,1) \bmod N \equiv 3975392$$

To decrypt (or decipher) the encrypted message, we calculate:

$$V_{24017}(3975392,1) \bmod N \equiv 11111$$

For a second example, let message = P = 22222. $D = P^2 - 4 = 1170499 \bmod 4071461$
Now calculate the Legendres of D with respect to p & q: (D/1949) = -+1; (D/2089) = -1. This means that the Lehmer totient to choose is the second in the above list, = LCM(1948, 2090) = 2035660. As above, we solve the modular equation $ed \equiv 1$ modulo r. d turns out to equal 55367.
To encrypt 22222, calculate $V_{1103}(22222,1) \bmod N \equiv 3055262$.
To decrypt, calculate $V_{55367}(3055262,1) \bmod N \equiv 22222$.

9.6 Name of Sponsoring Authority

Standards New Zealand
Private Bag 2439
Wellington 6020
New Zealand

9.7 Dates of Registration and Modifications 29 July 1994

9.8 Whether the Subject of a National Standard - No

9.9 Patent Licence Restrictions

LUC is the subject of U.S. Patent Application #07/953,832 "Cryptographic Communication Method and Apparatus" in the name of Peter John smith; and of New Zealand Patent Application 240019 "Cryptographic Communication method and Apparatus".

9.10 References

References are contained in the second section of the accompanying document "LUC Encryption Standard"..

9.11 Description of Algorithm

The LUC algorithm is described in the accompanying document "LUC Encryption Standard".

9.12 Modes of Operation

The modes of operation of LUC are described in the accompanying document "LUC Encryption Standard".

9.13 Other Information

Unsure of what to put here: one could write a book.

ISO9979/0003

LUC Encryption Standard

LUC Encryption Technology, Ltd.
25 Lawrence St, Herne Bay,
Auckland, New Zealand

Version 1.0

1. Scope

This standard describes a method for enciphering (or encrypting - the words are used interchangeably) data using the LUC public-key cryptosystem (or cipher) [1]. The primary uses of LUC will be to make either digital signatures or digital envelopes, although it can also be used to encipher short messages.

> For long digital signatures, the message (or content) is first hashed to a smaller string with a one-way hashing (message-digesting) algorithm (such as MD5 [2]). The hashed message string is then encrypted with the sender's secret LUC key. The signature is then appended to the message to give a digital signature.
>
> For digital envelopes, the message is first enciphered with a fast symmetric cipher (such as DEA or IDEAL), the symmetric cipher's key is encrypted with the LUC public key of the receiver. The enciphered message and key comprise the digital envelope.

This document also describes a syntax for LUC public and private keys. The public key syntax is the same as that in X.509 [3].

2. References

[1] P. J. Smith, M. J. J. Lennon, "LUC: A New Public Key System", IFIP/Sec '93, Proceedings of the Ninth IFIP International Symposium on Computer Security, Ontario, Canada, May 1993, Elsevier Science Publications, Netherlands, pp 97-111

[2] RSA Data Security, Inc. *PKCS #7: Cryptographic Message Syntax Standard*, version 1.4, June 1991.

[3] CCITT. *Recommendations X.509: The directory-Authentication Framework.* 1988.

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

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

3. Definitions

For the purposes of this standard there are the following definitions.

`AlgorithmIdentifier:` A type that identifies an algorithm (by object identifier) and any associated parameters, as laid out in X.509 [3].

ASN.1: Abstract Syntax Notation One, as defined in X.208 [4].

BER: Basic Encoding Rules for ASN.1, as laid out in X.209 [5].

modulus: Integer constructed as the product of two prime numbers.

private key: Modulus and four private subscripts.

public key: Modulus and public subscript.

4. Symbols and abbreviations

Upper-case italic symbols (e.g. *LUC*) denote octet strings and bit strings (in the case of the signature *S*); lower-case italic symbols (e.g. *b*) denote integers.

`ab`	hexadecimal octet value
BT	block type
D	data
	enciphering block
	enciphered data
M	message
PS	padding string
S	signature (a bit string)
	subscript (private or public)
d1, d2, d3, d4	private subscripts
e	public subscript
k	length of modulus in octets
n	modulus
	prime factors of modulus
x	integer encipherment block

y	integer enciphered data
mod n	arithmetic computation modulo n
	concatenation of octet strings X and Y
	length in octets of octet string X

5. General overview

The following four sections (6 through 9) specify LUC key generation, key syntax, the enciphering process, and the deciphering process. Each entity (sender and receiver) shall generate a pair of matching public and private keys. The enciphering process shall be done with one key and the deciphering done with the other. The enciphering can be done with either of the keys as long as deciphering is done with the other. In other words, LUC is symmetric with respect to enciphering and deciphering since one is the mathematical inverse of the other. Both processes transform an octet string to another octet string.

6. Key generation

Each entity shall select an odd positive integer e as its public exponent.

Privately each shall select two different odd prime numbers p and q such that $(p-1)$ and e shall have no common divisor apart from 1, and such that $(q-1)$ and e shall have no common divisor apart from 1.

The public modulus n shall be the product of the private prime numbers p and q.

$$n = pq$$

The private subscripts, *d1*, *d2*, *d3*, and *d4* are calculated in the following manner:

$$d1 \equiv e^{-1} \bmod (\mathrm{LCM}(p - 1), (q - 1))$$

$$d2 \equiv e^{-1} \bmod (\mathrm{LCM}(p - 1), (q + 1))$$

$$d3 \equiv e^{-1} \bmod (\mathrm{LCM}(p + 1), (q - 1))$$

$$d4 \equiv e^{-1} \bmod (\mathrm{LCM}(p + 1), (q + 1))$$

where LCM(a,b) denotes the least common multiple of a and b.

The length of the modulus n in octets is the integer k satisfying

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

For this standard, the length k of the modulus must be at least twelve octets.

Notes.

1. The public subscript, e, may be standardised in specific applications. The Fermat primes 3, 5, 17, 257, and 65537 are commonly chosen because they minimise the amount of calculation.

2. Additional conditions on the choice of primes should be taken into account. Neither of $(p + 1)$ and $(p - 1)$ nor $(q + 1)$ and $(q - 1)$ should be made up of small prime numbers. As well as guarding against various methods of factorisation, this constraint tends to maximise the message space.

7. Key syntax

This section gives the syntax for LUC public and private keys.

7.1 Public-key syntax

A LUC public key shall have ASN.1 type `LUCPublicKey`:

```
LUCPublicKey ::= SEQUENCE {
 modulus INTEGER, -- n
 publicSubscript INTEGER -- e }
```

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

`modulus` is the modulus n

`publicSubscript` is the public exponent e.

7.2 Private-key syntax

A LUC private key shall have ASN.1 type `LUCPrivateKey`:

```
LUCPrivateKey ::= SEQUENCE {
  version Version,
  modulus INTEGER, -- n
  publicSubscript INTEGER, -- e
  privateSubscript1 INTEGER, -- d1
  privateSubscript2 INTEGER, -- d2
  privateSubscript3 INTEGER, -- d3
  privateSubscript4 INTEGER, -- d4
  prime1 INTEGER, -- p
  prime2 INTEGER, -- q
  coefficient1 INTEGER, -- qi (inverse of q mod p)
  coefficient2 INTEGER -- pi (inverse of p mod q)

Version ::= INTEGER
```

The fields of `LUCPrivateKey` have the following meanings:

`version` is the version number, which is set at 0 for this version of the LUC Standard.

`modulus` is the modulus n.

`publicSubscript` is the public subscript e.

`privateSubscript1` is the private subscript *d1*.

`privateSubscript2` is the private subscript *d2*.

`privateSubscript3` is the private subscript *d3*.

`privateSubscript4` is the private subscript *d4*.

`prime1` is the prime factor *p* of *n*.

`prime1` is the prime factor *p* of *n*.

`coefficient1` is the Chinese Remainder Theorem coefficient *qi*, the multiplicative inverse of *q* mod *p*.

`coefficient2` is the Chinese Remainder Theorem coefficient *pi*, the multiplicative inverse of *p* mod *q*.

Note.

As far as the LUC algorithm is concerned, the only values needed for the private key are the modulus *n* and the four private subscripts *d1, d2, d3,* and *d4*. The additional values - *p, q, qi,* and *pi* - can be used to make the deciphering process more efficient. Given only the modulus *n*, the four private subscripts *d1, d2, d3,* and *d4*, and the public subscript *e*, these additional values can be derived.

8. Enciphering process

This section describes the LUC enciphering method.

Enciphering is done in four steps: enciphering block formatting, octet-string to integer conversion, LUC computation, and finally, integer to octet-string conversion. The input to thew enciphering process shall be the octet string *D*, the data; and integer *n*, the modulus; and an integer *c*, the subscript. For a public-key operation, the integer *c* shall be the entity's public subscript *e*; for a private-key operation, it shall be one of *d1, d2, d3,* or *d4*, the entity's private subscripts. The output from the enciphering process shall be an octet string *ED*, the enciphered data.

8.1 Enciphering block formatting

A block type *BT*, a padding string *PS*, and the data *D* shall be formatted into an octet string *EB*, the encryption block.

$$EB = 00 \| BT \| PS \| 00 \| D \qquad (1)$$

The block type *BT* shall be a single octet indicating the structure of the enciphering block. It has value `00`, `01` or `02`. For a private-key operation, the block type shall be `00` or `01`. For a public-key operation, it shall be `02`.

The padding string *PS* shall be made up of $k - 3 - \|D\|$ octets. For block type `00`, the octets shall have value `00`; for block type `01` they shall have value `FF`; and for block type `02` they

should be random, non-zero values. This makes the length of the enciphering block *EB* equal to *k*.

Notes.

1 For block type 00, the data *D* must begin with a non-zero octet or else be of known length so that the enciphering block can be parsed unambiguously. For block types 01 and 02, the enciphering block can be parsed unambiguously since the padding string *PS* contains no octets with value 00 and the padding string is separated from the data *D* by an octet with value 00.

2. Block type 01 is recommended for private-key operations. Block type 01 has the characteristic that the magnitude of the integer input to the LUC computation is large, which prevents certain cryptanalytic attacks.

3 For block type 02, it is recommended that the random, non-zero octets be generated independently for each enciphering operation, especially if the same data is input to more than one enciphering operation.

8.2 Octet string to integer conversion

The enciphering block *EB* shall be converted to an integer *x*, the integer enciphering block. Let $EB_1,\ldots, EB_k$ be the k octets of *EB*. Then *x* should satisfy the following equality:

$$x = \sum_{j=1}^{k} 2^{8(k-j)} EB_j \qquad (2)$$

Consequently, the first octet of *EB* is the most significant in the integer and the last octet of *EB* is the least significant.

Note.

The integer enciphering block *x* satisfies the inequalities $0 \le x < n$, since $EB_1 = 00$ and $2^{8(k-1)} \le n$.

8.3 LUC computation

The integer enciphering block *x* shall give the integer *y* by calculating the *c*th Lucas function modulo *n*.

$$y = V_c(x,1) \bmod n,\ 0 \le y < n$$

This is the canonical LUC computation.

8.4 Integer to octet string conversion

The integer enciphered data y shall be converted to an octet string *ED* of length *k*. The encrypted data shall satisfy

$$y = \sum_{j=1}^{k} 2^{8(k-j)} ED_j \qquad (3)$$

where ED_1,..., ED_k are the octets of *ED* from most significant to least significant.

9. Deciphering process

This section describes the LUC deciphering process.

Deciphering is made up of four steps: octet string to integer conversion, LUC computation, integer to octet string conversion, and enciphering block parsing. The input to the deciphering process shall be an octet string *ED*, the enciphered data; an integer *n*, the modulus; and an integer *c*, the subscript.

For a public-key operation, the integer *c* shall be an entity's public subscript *e*; for a private-key operation, it shall be chosen by the following process: If $(y^2 - 4)$ is a quadratic residue of prime *p* and a quadratic residue of prime *q* then the deciphering subscript is *d1*; if $(y^2 - 4)$ is a quadratic residue of prime *p* and a quadratic non-residue of prime *q* then the deciphering subscript is *d2*; if $(y^2 - 4)$ is a quadratic non-residue of prime *p* and a quadratic residue of prime *q* then the deciphering subscript is *d3*; and if $(y^2 - 4)$ is a quadratic non-residue of prime *p* and a quadratic non-residue of prime *q* then the deciphering subscript is *d4*.

The output from the deciphering process shall be an octet string *D*, the data.

9.1 Octet string to integer conversion

The enciphered data *ED* shall be converted to an integer *y*, the integer enciphered data, according to Equation (3).

9.2 LUC computation

The integer enciphered data *y* shall give the integer *x* by calculating the *c*th Lucas function modulo *n*.

$$x = V_c(y,1) \bmod n,\ 0 \le x < n$$

This is the canonical LUC computation.

9.3 Integer to octet string conversion

The integer encryption block *x* shall be converted to an octet string *EB* of length *k*, the encryption block, according to Equation (2).

9.4 Encryption block parsing

The encryption block *EB* shall be parsed into a block type *BT*, a padding string *PS*, and the data *D* according to Equation (1).

It is an error if any of the following conditions occurs:

The encryption block *EB* cannot be parsed unambiguously;

The padding string *PS* consists of fewer than eight octets, or is inconsistent with the block type *BT*; or

The deciphering process is a public-key operation and the block type *BT* is not 00 or 01, or the deciphering process is a private-key operation and the block type is not 02.