Internet Draft Peter Gutmann
draft-rfced-info-gutmann-00.txt
June 16, 1997
Expires December, 1997

# Description of the EP2 Cipher
<draft-rfced-info-gutmann-00.txt>

Status of this memo



## 1. Introduction

The EP2 cipher is a block cipher that is useful in many cryptographic applications. It is believed to be interoperable with the RC2 cipher from RSA Data Security, Inc., which has been specified for use in many Internet protocols.

## 2. Description

The EP2 cipher is word oriented, operating on a block of 64 bits divided into four 16-bit words, with a key table of 64 words. All data units are little-endian. This functional description of the algorithm is based on [RC5], using the same general layout, terminology, and pseudocode style.

## 3. Notation and Primitive Operations

EP2 uses the following primitive operations:

1. **Two's-complement addition of words, denoted by "+".** The inverse operation, subtraction, is denoted by "-".
2. **Bitwise exclusive OR, denoted by "^".**
3. **Bitwise AND, denoted by "&".**
4. **Bitwise NOT, denoted by "~".**
5. **A left-rotation of words; the rotation of word x left by y is**

**denoted by "x <<< y". The inverse operation, right-rotation,** is denoted by "x >>> y".

These operations are directly and efficiently supported by most processors.

## 3. EP2 Algorithm

EP2 consists of three components, a key expansion algorithm, an encryption algorithm, and a decryption algorithm.

### 3.1 Key Expansion

The purpose of the key-expansion routine is to expand the user's key K to fill the expanded key array S, so S resembles an array of random binary words determined by the user's secret key K.

#### 3.1.1 Initialising the S-box

EP2 uses a single 256-byte S-box derived from the ciphertext contents of Beale Cipher No.1 XOR'd with a one-time pad. The Beale Ciphers predate modern cryptography by enough time that there should be no concerns about trapdoors hidden in the data. They have been published widely, and the S-box can be easily recreated from the one-time pad values and the Beale Cipher data taken from a standard source. To initialise the S-box:

```
for i = 0 to 255 do
  sBox[ i ] = ( beale[ i ] mod 256 ) ^ pad[ i ]
```

The contents of Beale Cipher No.1 and the necessary one-time pad are given as an appendix at the end of this document. For efficiency, implementors may wish to skip the Beale Cipher expansion and store the sBox table directly.

#### 3.1.2 Expanding the Secret Key to 128 Bytes

The secret key is first expanded to fill 128 bytes (64 words). The expansion consists of taking the sum of the first and last bytes in the user key, looking up the sum (modulo 256) in the S-box, and appending the result to the key. The operation is repeated with the second byte and new last byte of the key until all 128 bytes have been generated. Note that the following pseudocode treats the S array as an array of 128 bytes rather than 64 words.

```
for j = 0 to length-1 do
  S[ j ] = K[ j ]
for j = length to 127 do
  s[ j ] = sBox[ ( S[ j-length ] + S[ j-1 ] ) mod 256 ]
```

#### 3.1.3 Reducing the Effective Key Length

At this point it is possible to perform a truncation of the effective key length to ease the creation of espionage-enabled software products. To use a key with an effective size of 'reducedLength' bytes, the following transformation is used.

```
  maxValue = 128 - reducedLength
  S[ maxValue ] = sBox[ S[ maxValue ] ]
  for j = maxValue - 1 to 0 step -1 do
    S[ j ] = sBox[ S[ j + 1 ] ^ S[ j + len ] ]
```

For example to reduce a key to an effective size of 40 bits the transformation is:

```
  S[ 88 ] = sBox[ S[ 88 ] ]
  for j = 87 to 0 step -1 do
    S[ j ] = sBox[ S[ j + 1 ] ^ S[ j + len ] ]
```

If no reduction of effective keysize is required, the above can be simplified to replacing the first byte of S with the entry selected from the S-box:

```
  S[ 0 ] = sBox[ S[ 0 ] ]
```

### 3.2 Encryption

The cipher has 16 full rounds, each divided into 4 subrounds. Two of the full rounds perform an additional transformation on the data. Note that the following pseudocode treats the S array as an array of **64 words rather than 128 bytes.**

```
  for i = 0 to 15 do
    j = i * 4;
    word0 = ( word0 + ( word1 & ~word3 ) +
       ( word2 & word3 ) + S[ j+0 ] ) <<< 1
    word1 = ( word1 + ( word2 & ~word0 ) +
       ( word3 & word0 ) + S[ j+1 ] ) <<< 2
    word2 = ( word2 + ( word3 & ~word1 ) +
       ( word0 & word1 ) + S[ j+2 ] ) <<< 3
    word3 = ( word3 + ( word0 & ~word2 ) +
       ( word1 & word2 ) + S[ j+3 ] ) <<< 5
```

In addition, the fifth and eleventh rounds add the contents of the S-box indexed by one of the data words to another of the data words following the four subrounds as follows:

```
    word0 = word0 + S[ word3 & 63 ];
    word1 = word1 + S[ word0 & 63 ];
    word2 = word2 + S[ word1 & 63 ];
    word3 = word3 + S[ word2 & 63 ];
```

### 3.3 Decryption

The decryption operation is simply the inverse of the encryption operation. Note that the following pseudocode treats the S array as an array of 64 words rather than 128 bytes.

```
  for i = 15 downto 0 do
    j = i * 4;
    word3 = ( word3 >>> 5 ) - ( word0 & ~word2 ) -
       ( word1 & word2 ) - S[ j+3 ]
    word2 = ( word2 >>> 3 ) - ( word3 & ~word1 ) -
       ( word0 & word1 ) - S[ j+2 ]
    word1 = ( word1 >>> 2 ) - ( word2 & ~word0 ) -
       ( word3 & word0 ) - S[ j+1 ]
    word0 = ( word0 >>> 1 ) - ( word1 & ~word3 ) -
       ( word2 & word3 ) - S[ j+0 ]
```

In addition, the fifth and eleventh rounds subtract the contents of the S-box indexed by one of the data words from another one of the data words following the four subrounds as follows:

```
    word3 = word3 - S[ word2 & 63 ]
    word2 = word2 - S[ word1 & 63 ]
    word1 = word1 - S[ word0 & 63 ]
    word0 = word0 - S[ word3 & 63 ]
```

## 4. Test Vectors

The following test vectors may be used to test the correctness of an EP2 implementation:

```
  Key:      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  Plain:    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  Cipher:   0x1C, 0x19, 0x8A, 0x83, 0x8D, 0xF0, 0x28, 0xB7

  Key:      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01
  Plain:    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  Cipher:   0x21, 0x82, 0x9C, 0x78, 0xA9, 0xF9, 0xC0, 0x74

  Key:      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  Plain:    0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
  Cipher:   0x13, 0xDB, 0x35, 0x17, 0xD3, 0x21, 0x86, 0x9E

  Key:      0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
            0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F
  Plain:    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
  Cipher:   0x50, 0xDC, 0x01, 0x62, 0xBD, 0x75, 0x7F, 0x31
```

The following ciphertext is produced from the first key/plaintext

combination given above if the 128-bit effective key length is reduced to the given lengths using the algorithm in section 3.1.3:

| Effective key Bits | Bytes | Ciphertext |
|---|---|---|
| 40 | 5 | 0x65, 0x8A, 0x83, 0x3A, 0x5D, 0xE3, 0x45, 0x55 |
| 48 | 6 | 0x94, 0x42, 0x96, 0x80, 0xD5, 0xD6, 0xFE, 0xD2 |
| 56 | 7 | 0xD0, 0xDC, 0x8D, 0x97, 0xB3, 0x2C, 0xC8, 0xB7 |
| 64 | 8 | 0x93, 0xCC, 0x73, 0xC9, 0xF7, 0x4E, 0x32, 0x82 |

## 5. Security

This paper was first widely published in early 1996, and there have been no known successful attacks on the algorithm since then. Further, there have been no known successful attacks on the RC2 algorithm, with which the algorithm described in this paper is thought to be interoperable.

## A. Beale Cipher No.1, "The Locality of the Vault"

Beale Cipher No.1.

71, 194, 38,1701, 89, 76, 11, 83, 1629, 48, 94, 63, 132, 16, 111, 95,
84, 341, 975, 14, 40, 64, 27, 81, 139, 213, 63, 90,1120, 8, 15, 3,
126, 2018, 40, 74, 758, 485, 604, 230, 436, 664, 582, 150, 251, 284,
308, 231, 124, 211, 486, 225, 401, 370, 11, 101, 305, 139, 189, 17,
33, 88, 208, 193, 145, 1, 94, 73, 416, 918, 263, 28, 500, 538, 356,
117, 136, 219, 27, 176, 130, 10, 460, 25, 485, 18, 436, 65, 84, 200,
283, 118, 320, 138, 36, 416, 280, 15, 71, 224, 961, 44, 16, 401, 39,
88, 61, 304, 12, 21, 24, 283, 134, 92, 63, 246, 486, 682, 7, 219, 184,
360, 780, 18, 64, 463, 474, 131, 160, 79, 73, 440, 95, 18, 64, 581,
34, 69, 128, 367, 460, 17, 81, 12, 103, 820, 62, 110, 97, 103, 862,
70, 60, 1317, 471, 540, 208, 121, 890, 346, 36, 150, 59, 568, 614, 13,
120, 63, 219, 812, 2160, 1780, 99, 35, 18, 21, 136, 872, 15, 28, 170,
88, 4, 30, 44, 112, 18, 147, 436, 195, 320, 37, 122, 113, 6, 140, 8,
120, 305, 42, 58, 461, 44, 106, 301, 13, 408, 680, 93, 86, 116, 530,
82, 568, 9, 102, 38, 416, 89, 71, 216, 728, 965, 818, 2, 38, 121, 195,
14, 326, 148, 234, 18, 55, 131, 234, 361, 824, 5, 81, 623, 48, 961,
19, 26, 33, 10, 1101, 365, 92, 88, 181, 275, 346, 201, 206

## B. One-time Pad for Creating the S-Box

158, 186, 223, 97, 64, 145, 190, 190, 117, 217, 163, 70, 206, 176,
183, 194, 146, 43, 248, 141, 3, 54, 72, 223, 233, 153, 91, 210, 36,
131, 244, 161, 105, 120, 113, 191, 113, 86, 19, 245, 213, 221, 43, 27,
242, 157, 73, 213, 193, 92, 166, 10, 23, 197, 112, 110, 193, 30, 156,
51, 125, 51, 158, 67, 197, 215, 59, 218, 110, 246, 181, 0, 135, 76,
164, 97, 47, 87, 234, 108, 144, 127, 6, 6, 222, 172, 80, 144, 22, 245,
207, 70, 227, 182, 146, 134, 119, 176, 73, 58, 135, 69, 23, 198, 0,
170, 32, 171, 176, 129, 91, 24, 126, 77, 248, 0, 118, 69, 57, 60, 190,

171, 217, 61, 136, 169, 196, 84, 168, 167, 163, 102, 223, 64, 174,
178, 166, 239, 242, 195, 249, 92, 59, 38, 241, 46, 236, 31, 59, 114,
23, 50, 119, 186, 7, 66, 212, 97, 222, 182, 230, 118, 122, 86, 105,
92, 179, 243, 255, 189, 223, 164, 194, 215, 98, 44, 17, 20, 53, 153,
137, 224, 176, 100, 208, 114, 36, 200, 145, 150, 215, 20, 87, 44, 252,
20, 235, 242, 163, 132, 63, 18, 5, 122, 74, 97, 34, 97, 142, 86, 146,
221, 179, 166, 161, 74, 69, 182, 88, 120, 128, 58, 76, 155, 15, 30,
77, 216, 165, 117, 107, 90, 169, 127, 143, 181, 208, 137, 200, 127,
170, 195, 26, 84, 255, 132, 150, 58, 103, 250, 120, 221, 237, 37, 8,
99

## C. References

[RC5] Ron Rivest, "The RC5 Encryption Algorithm", Proceedings of the
Second International Workshop on Fast Software Encryption,
Springer-Verlag LNCS No.1008.

## D. Author's Address

Peter Gutmann
University of Auckland
Private Bag 92019
Auckland
New Zealand

Phone: +64 9 373-7599
Email: pgut001@cs.auckland.ac.nz