       Raptor Forward Error Correction Scheme for Object Delivery

Status of This Memo

Abstract

   This document describes a Fully-Specified Forward Error Correction
   (FEC) scheme, corresponding to FEC Encoding ID 1, for the Raptor
   forward error correction code and its application to reliable
   delivery of data objects.

   Raptor is a fountain code, i.e., as many encoding symbols as needed
   can be generated by the encoder on-the-fly from the source symbols of
   a source block of data. The decoder is able to recover the source
   block from any set of encoding symbols only slightly more in number
   than the number of source symbols.

   The Raptor code described here is a systematic code, meaning that all
   the source symbols are among the encoding symbols that can be
   generated.

Table of Contents

1. Introduction

   This document specifies an FEC Scheme for the Raptor forward error
   correction code for object delivery applications. The concept of an
   FEC Scheme is defined in [RFC5052] and this document follows the
   format prescribed there and uses the terminology of that document.
   Raptor Codes were introduced in [Raptor]. For an overview, see, for
   example, [CCNC].

   The Raptor FEC Scheme is a Fully-Specified FEC Scheme corresponding
   to FEC Encoding ID 1.

   Raptor is a fountain code, i.e., as many encoding symbols as needed
   can be generated by the encoder on-the-fly from the source symbols of
   a block. The decoder is able to recover the source block from any
   set of encoding symbols only slightly more in number than the number
   of source symbols.

   The code described in this document is a systematic code, that is,
   the original source symbols can be sent unmodified from sender to
   receiver, as well as a number of repair symbols. For more background
   on the use of Forward Error Correction codes in reliable multicast,
   see [RFC3453].

   The code described here is identical to that described in [MBMS].

2. Requirements Notation

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in [RFC2119].

3. Formats and Codes

3.1. FEC Payload IDs

   The FEC Payload ID MUST be a 4 octet field defined as follows:

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |      Source Block Number      |       Encoding Symbol ID      |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
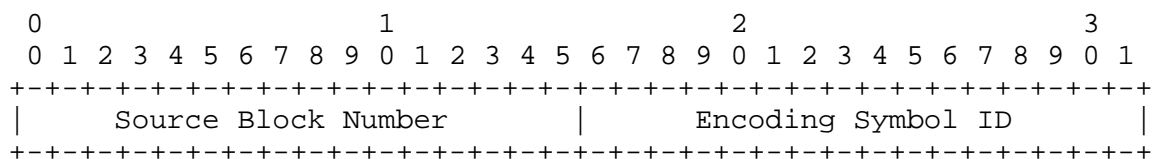
                      Figure 1: FEC Payload ID format

Source Block Number (SBN), (16 bits): An integer identifier for
the source block that the encoding symbols within the packet
relate to.

Encoding Symbol ID (ESI), (16 bits): An integer identifier for the
encoding symbols within the packet.

The interpretation of the Source Block Number and Encoding Symbol
Identifier is defined in Section 5.

3.2. FEC Object Transmission Information (OTI)

3.2.1. Mandatory

The value of the FEC Encoding ID MUST be 1 (one), as assigned by IANA
(see Section 7).

3.2.2. Common

The Common FEC Object Transmission Information elements used by this
FEC Scheme are:

   - Transfer Length (F)

   - Encoding Symbol Length (T)

The Transfer Length is a non-negative integer less than 2^{45}. The
Encoding Symbol Length is a non-negative integer less than 2^{16}.

The encoded Common FEC Object Transmission Information format is
shown in Figure 2.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Transfer Length                         |
   +                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |            Reserved           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Encoding Symbol Length    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
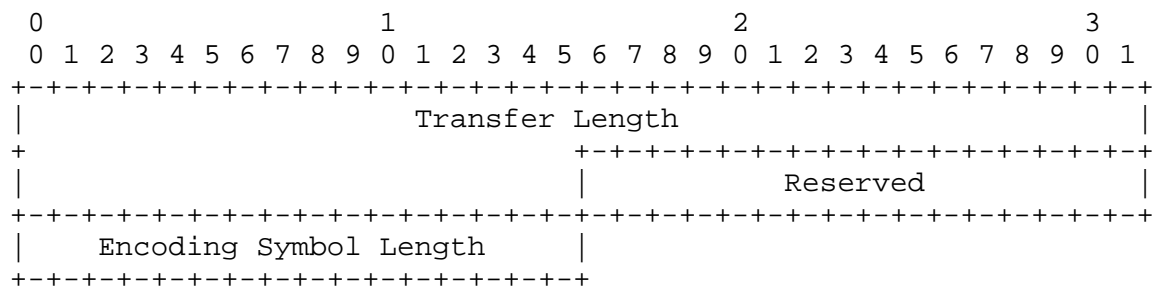
           Figure 2: Encoded Common FEC OTI for Raptor FEC Scheme

NOTE 1: The limit of 2^{45} on the transfer length is a consequence
of the limitation on the symbol size to $2^{16}-1$, the limitation on
the number of symbols in a source block to 2^{13}, and the

limitation on the number of source blocks to 2^^16. However, the
Transfer Length is encoded as a 48-bit field for simplicity.

3.2.3. Scheme-Specific

The following parameters are carried in the Scheme-Specific FEC
Object Transmission Information element for this FEC Scheme:

   - The number of source blocks (Z)

   - The number of sub-blocks (N)

   - A symbol alignment parameter (Al)

These parameters are all non-negative integers. The encoded Scheme-
specific Object Transmission Information is a 4-octet field
consisting of the parameters Z (2 octets), N (1 octet), and Al (1
octet) as shown in Figure 3.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Z               |       N       |      Al       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
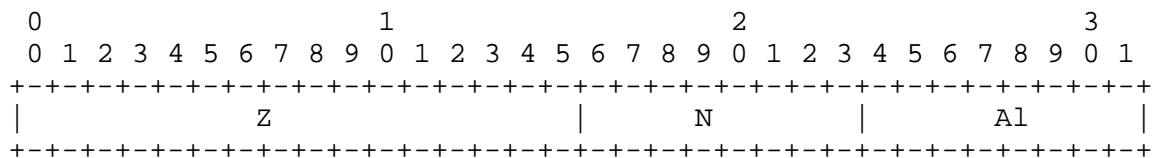
Figure 3: Encoded Scheme-Specific FEC Object Transmission Information

The encoded FEC Object Transmission Information is a 14-octet field
consisting of the concatenation of the encoded Common FEC Object
Transmission Information and the encoded Scheme-Specific FEC Object
Transmission Information.

These three parameters define the source block partitioning as
described in Section 5.3.1.2.

4. Procedures

4.1. Content Delivery Protocol Requirements

This section describes the information exchange between the Raptor
FEC Scheme and any Content Delivery Protocol (CDP) that makes use of
the Raptor FEC Scheme for object delivery.

The Raptor encoder and decoder for object delivery require the
following information from the CDP:

   - The transfer length of the object, F, in bytes

- A symbol alignment parameter, Al

- The symbol size, T, in bytes, which MUST be a multiple of Al

- The number of source blocks, Z

- The number of sub-blocks in each source block, N

The Raptor encoder for object delivery additionally requires:

- the object to be encoded, F bytes

The Raptor encoder supplies the CDP with the following information
for each packet to be sent:

- Source Block Number (SBN)

- Encoding Symbol ID (ESI)

- Encoding symbol(s)

The CDP MUST communicate this information to the receiver.

4.2. Example Parameter Derivation Algorithm

This section provides recommendations for the derivation of the three
transport parameters, T, Z, and N. This recommendation is based on
the following input parameters:

- F   the transfer length of the object, in bytes

- W   a target on the sub-block size, in bytes

- P   the maximum packet payload size, in bytes, which is assumed to
      be a multiple of Al

- Al the symbol alignment parameter, in bytes

- Kmax   the maximum number of source symbols per source block.

        Note: Section 5.1.2 defines Kmax to be 8192.

- Kmin   a minimum target on the number of symbols per source block

- Gmax   a maximum target number of symbols per packet

   Based on the above inputs, the transport parameters T, Z, and N are
   calculated as follows:

   Let

      G = min{ceil(P*Kmin/F), P/Al, Gmax}

      T = floor(P/(Al*G))*Al

      Kt = ceil(F/T)

      Z = ceil(Kt/Kmax)

      N = min{ceil(ceil(Kt/Z)*T/W), T/Al}

   The value G represents the maximum number of symbols to be
   transported in a single packet. The value Kt is the total number of
   symbols required to represent the source data of the object. The
   values of G and N derived above should be considered as lower bounds.
   It may be advantageous to increase these values, for example, to the
   nearest power of two. In particular, the above algorithm does not
   guarantee that the symbol size, T, divides the maximum packet size,
   P, and so it may not be possible to use the packets of size exactly
   P. If, instead, G is chosen to be a value that divides P/Al, then
   the symbol size, T, will be a divisor of P and packets of size P can
   be used.

   The algorithm above and that defined in Section 5.3.1.2 ensure that
   the sub-symbol sizes are a multiple of the symbol alignment
   parameter, Al. This is useful because the XOR operations used for
   encoding and decoding are generally performed several bytes at a
   time, for example, at least 4 bytes at a time on a 32-bit processor.
   Thus, the encoding and decoding can be performed faster if the sub-
   symbol sizes are a multiple of this number of bytes.

   Recommended settings for the input parameters, Al, Kmin, and Gmax are
   as follows: Al = 4, Kmin = 1024, Gmax = 10.

   The parameter W can be used to generate encoded data that can be
   decoded efficiently with limited working memory at the decoder. Note
   that the actual maximum decoder memory requirement for a given value
   of W depends on the implementation, but it is possible to implement
   decoding using working memory only slightly larger than W.

5. Raptor FEC Code Specification

5.1. Definitions, Symbols, and Abbreviations

5.1.1. Definitions

   For the purposes of this specification, the following terms and
   definitions apply.

      Source block: a block of K source symbols that are considered
      together for Raptor encoding purposes.

      Source symbol: the smallest unit of data used during the encoding
      process. All source symbols within a source block have the same
      size.

      Encoding symbol: a symbol that is included in a data packet. The
      encoding symbols consist of the source symbols and the repair
      symbols. Repair symbols generated from a source block have the
      same size as the source symbols of that source block.

      Systematic code: a code in which all the source symbols may be
      included as part of the encoding symbols sent for a source block.

      Repair symbol: the encoding symbols sent for a source block that
      are not the source symbols. The repair symbols are generated
      based on the source symbols.

      Intermediate symbols: symbols generated from the source symbols
      using an inverse encoding process . The repair symbols are then
      generated directly from the intermediate symbols. The encoding
      symbols do not include the intermediate symbols, i.e.,
      intermediate symbols are not included in data packets.

      Symbol: a unit of data. The size, in bytes, of a symbol is known
      as the symbol size.

      Encoding symbol group: a group of encoding symbols that are sent
      together, i.e., within the same packet whose relationship to the
      source symbols can be derived from a single Encoding Symbol ID.

      Encoding Symbol ID: information that defines the relationship
      between the symbols of an encoding symbol group and the source
      symbols.

      Encoding packet: data packets that contain encoding symbols

Sub-block: a source block is sometimes broken into sub-blocks,
each of which is sufficiently small to be decoded in working
memory. For a source block consisting of K source symbols, each
sub-block consists of K sub-symbols, each symbol of the source
block being composed of one sub-symbol from each sub-block.

Sub-symbol: part of a symbol. Each source symbol is composed of
as many sub-symbols as there are sub-blocks in the source block.

Source packet: data packets that contain source symbols.

Repair packet: data packets that contain repair symbols.

5.1.2. Symbols

i, j, x, h, a, b, d, v, m represent positive integers.

ceil(x) denotes the smallest positive integer that is greater than
        or equal to x.

choose(i,j) denotes the number of ways j objects can be chosen from
            among i objects without repetition.

floor(x) denotes the largest positive integer that is less than or
        equal to x.

i % j denotes i modulo j.

X ^ Y denotes, for equal-length bit strings X and Y, the bitwise
        exclusive-or of X and Y.

Al   denotes a symbol alignment parameter. Symbol and sub-symbol
      sizes are restricted to be multiples of Al.

A   denotes a matrix over GF(2).

Transpose[A] denotes the transposed matrix of matrix A.

A^^-1 denotes the inverse matrix of matrix A.

K   denotes the number of symbols in a single source block.

Kmax denotes the maximum number of source symbols that can be in a
     single source block. Set to 8192.

L   denotes the number of pre-coding symbols for a single source
     block.

S   denotes the number of LDPC symbols for a single source block.

H   denotes the number of Half symbols for a single source block.

C   denotes an array of intermediate symbols, C[0], C[1], C[2],...,
        C[L-1].

C'   denotes an array of source symbols, C'[0], C'[1], C'[2],...,
        C'[K-1].

X   a non-negative integer value

V0, V1 two arrays of 4-byte integers, V0[0], V0[1],..., V0[255] and
        V1[0], V1[1],..., V1[255]

Rand[X, i, m] a pseudo-random number generator

Deg[v] a degree generator

LTEnc[K, C ,(d, a, b)] a LT encoding symbol generator

Trip[K, X] a triple generator function

G   the number of symbols within an encoding symbol group

GF(n)   the Galois field with n elements.

N   the number of sub-blocks within a source block

T   the symbol size in bytes. If the source block is partitioned
        into sub-blocks, then T = T'*N.

T'   the sub-symbol size, in bytes. If the source block is not
        partitioned into sub-blocks, then T' is not relevant.

F   the transfer length of an object, in bytes

I   the sub-block size in bytes

P   for object delivery, the payload size of each packet, in bytes,
        that is used in the recommended derivation of the object
        delivery transport parameters.

Q   Q = 65521, i.e., Q is the largest prime smaller than 2^{16}

Z   the number of source blocks, for object delivery

J(K) the systematic index associated with K

   I_S denotes the SxS identity matrix.

   O_SxH denotes the SxH zero matrix.

   a ^^ b a raised to the power b

5.1.3. Abbreviations

   For the purposes of the present document, the following abbreviations
   apply:

   ESI   Encoding Symbol ID

   LDPC   Low Density Parity Check

   LT   Luby Transform

   SBN   Source Block Number

   SBL   Source Block Length (in units of symbols)

5.2. Overview

   The principal component of the systematic Raptor code is the basic
   encoder described in Section 5.4. First, it is described how to
   derive values for a set of intermediate symbols from the original
   source symbols such that knowledge of the intermediate symbols is
   sufficient to reconstruct the source symbols. Secondly, the encoder
   produces repair symbols, which are each the exclusive OR of a number
   of the intermediate symbols. The encoding symbols are the
   combination of the source and repair symbols. The repair symbols are
   produced in such a way that the intermediate symbols, and therefore
   also the source symbols, can be recovered from any sufficiently large
   set of encoding symbols.

   This document specifies the systematic Raptor code encoder. A number
   of possible decoding algorithms are possible. An efficient decoding
   algorithm is provided in Section 5.5.

   The construction of the intermediate and repair symbols is based in
   part on a pseudo-random number generator described in
   Section 5.4.4.1. This generator is based on a fixed set of 512
   random numbers that MUST be available to both sender and receiver.
   These are provided in Section 5.6.

Finally, the construction of the intermediate symbols from the source
symbols is governed by a 'systematic index', values of which are
provided in Section 5.7 for source block sizes from 4 source symbols
to Kmax = 8192 source symbols.

5.3. Object Delivery

5.3.1. Source Block Construction

5.3.1.1. General

In order to apply the Raptor encoder to a source object, the object
may be broken into Z >= 1 blocks, known as source blocks. The Raptor
encoder is applied independently to each source block. Each source
block is identified by a unique integer Source Block Number (SBN),
where the first source block has SBN zero, the second has SBN one,
etc. Each source block is divided into a number, K, of source
symbols of size T bytes each. Each source symbol is identified by a
unique integer Encoding Symbol Identifier (ESI), where the first
source symbol of a source block has ESI zero, the second has ESI one,
etc.

Each source block with K source symbols is divided into N >= 1 sub-
blocks, which are small enough to be decoded in the working memory.
Each sub-block is divided into K sub-symbols of size T'.

Note that the value of K is not necessarily the same for each source
block of an object and the value of T' may not necessarily be the
same for each sub-block of a source block. However, the symbol size
T is the same for all source blocks of an object and the number of
symbols, K, is the same for every sub-block of a source block. Exact
partitioning of the object into source blocks and sub-blocks is
described in Section 5.3.1.2 below.

5.3.1.2. Source Block and Sub-Block Partitioning

The construction of source blocks and sub-blocks is determined based
on five input parameters, F, Al, T, Z, and N, and a function
Partition[]. The five input parameters are defined as follows:

- F the transfer length of the object, in bytes

- Al a symbol alignment parameter, in bytes

- T the symbol size, in bytes, which MUST be a multiple of Al

- Z the number of source blocks

- N the number of sub-blocks in each source block

These parameters MUST be set so that ceil(ceil(F/T)/Z) <= Kmax.
Recommendations for derivation of these parameters are provided in
Section 4.2.

The function Partition[] takes a pair of integers (I, J) as input and
derives four integers (IL, IS, JL, JS) as output. Specifically, the
value of Partition[I, J] is a sequence of four integers (IL, IS, JL,
JS), where IL = ceil(I/J), IS = floor(I/J), JL = I - IS * J, and JS =
J - JL. Partition[] derives parameters for partitioning a block of
size I into J approximately equal-sized blocks. Specifically, JL
blocks of length IL and JS blocks of length IS.

The source object MUST be partitioned into source blocks and sub-
blocks as follows:

Let

   Kt = ceil(F/T)

   (KL, KS, ZL, ZS) = Partition[Kt, Z]

   (TL, TS, NL, NS) = Partition[T/Al, N]

Then, the object MUST be partitioned into Z = ZL + ZS contiguous
source blocks, the first ZL source blocks each having length KL*T
bytes, and the remaining ZS source blocks each having KS*T bytes.

If Kt*T > F, then for encoding purposes, the last symbol MUST be
padded at the end with Kt*T - F zero bytes.

Next, each source block MUST be divided into N = NL + NS contiguous
sub-blocks, the first NL sub-blocks each consisting of K contiguous
sub-symbols of size of TL*Al and the remaining NS sub-blocks each
consisting of K contiguous sub-symbols of size of TS*Al. The symbol
alignment parameter Al ensures that sub-symbols are always a multiple
of Al bytes.

Finally, the m-th symbol of a source block consists of the
concatenation of the m-th sub-symbol from each of the N sub-blocks.
Note that this implies that when N > 1, then a symbol is NOT a
contiguous portion of the object.

5.3.2. Encoding Packet Construction

   Each encoding packet contains the following information:

      - Source Block Number (SBN)

      - Encoding Symbol ID (ESI)

      - encoding symbol(s)

   Each source block is encoded independently of the others. Source
   blocks are numbered consecutively from zero.

   Encoding Symbol ID values from 0 to K-1 identify the source symbols
   of a source block in sequential order, where K is the number of
   symbols in the source block. Encoding Symbol IDs from K onwards
   identify repair symbols.

   Each encoding packet either consists entirely of source symbols
   (source packet) or entirely of repair symbols (repair packet). A
   packet may contain any number of symbols from the same source block.
   In the case that the last source symbol in a source packet includes
   padding bytes added for FEC encoding purposes, then these bytes need
   not be included in the packet. Otherwise, only whole symbols MUST be
   included.

   The Encoding Symbol ID, X, carried in each source packet is the
   Encoding Symbol ID of the first source symbol carried in that packet.
   The subsequent source symbols in the packet have Encoding Symbol IDs,
   X+1 to X+G-1, in sequential order, where G is the number of symbols
   in the packet.

   Similarly, the Encoding Symbol ID, X, placed into a repair packet is
   the Encoding Symbol ID of the first repair symbol in the repair
   packet and the subsequent repair symbols in the packet have Encoding
   Symbol IDs X+1 to X+G-1 in sequential order, where G is the number of
   symbols in the packet.

   Note that it is not necessary for the receiver to know the total
   number of repair packets.

   Associated with each symbol is a triple of integers (d, a, b).

   The G repair symbol triples (d[0], a[0], b[0]),..., (d[G-1], a[G-1],
   b[G-1]) for the repair symbols placed into a repair packet with ESI X
   are computed using the Triple generator defined in Section 5.4.4.4 as
   follows:

      For each i = 0, ..., G-1, (d[i], a[i], b[i]) = Trip[K,X+i]

   The G repair symbols to be placed in repair packet with ESI X are
   calculated based on the repair symbol triples, as described in
   Section 5.4, using the intermediate symbols C and the LT encoder
   LTEnc[K, C, (d[i], a[i], b[i])].

5.4. Systematic Raptor Encoder

5.4.1. Encoding Overview

   The systematic Raptor encoder is used to generate repair symbols from
   a source block that consists of K source symbols.

   Symbols are the fundamental data units of the encoding and decoding
   process. For each source block (sub-block), all symbols (sub-
   symbols) are the same size. The atomic operation performed on
   symbols (sub-symbols) for both encoding and decoding is the
   exclusive-or operation.

   Let C'[0],..., C'[K-1] denote the K source symbols.

   Let C[0],..., C[L-1] denote L intermediate symbols.

   The first step of encoding is to generate a number, L > K, of
   intermediate symbols from the K source symbols. In this step, K
   source symbol triples (d[0], a[0], b[0]), ..., (d[K-1], a[K-1],
   b[K-1]) are generated using the Trip[] generator as described in
   Section 5.4.2.2. The K source symbol triples are associated with the
   K source symbols and are then used to determine the L intermediate
   symbols C[0],..., C[L-1] from the source symbols using an inverse
   encoding process. This process can be realized by a Raptor decoding
   process.

   Certain "pre-coding relationships" MUST hold within the L
   intermediate symbols. Section 5.4.2.3 describes these relationships
   and how the intermediate symbols are generated from the source
   symbols.

   Once the intermediate symbols have been generated, repair symbols are
   produced and one or more repair symbols are placed as a group into a
   single data packet. Each repair symbol group is associated with an
   Encoding Symbol ID (ESI) and a number, G, of repair symbols. The ESI
   is used to generate a triple of three integers, (d, a, b) for each
   repair symbol, again using the Trip[] generator as described in
   Section 5.4.4.4. Then, each (d,a,b)-triple is used to generate the

corresponding repair symbol from the intermediate symbols using the
LTEnc[K, C[0],..., C[L-1], (d,a,b)] generator described in
Section 5.4.4.3.

5.4.2. First Encoding Step: Intermediate Symbol Generation

5.4.2.1. General

The first encoding step is a pre-coding step to generate the L
intermediate symbols C[0], ..., C[L-1] from the source symbols C'[0],
..., C'[K-1]. The intermediate symbols are uniquely defined by two
sets of constraints:

   1. The intermediate symbols are related to the source symbols by
   a set of source symbol triples. The generation of the source
   symbol triples is defined in Section 5.4.2.2 using the Trip[]
   generator described in Section 5.4.4.4.

   2. A set of pre-coding relationships hold within the intermediate
   symbols themselves. These are defined in Section 5.4.2.3.

The generation of the L intermediate symbols is then defined in
Section 5.4.2.4

5.4.2.2. Source Symbol Triples

Each of the K source symbols is associated with a triple (d[i], a[i],
b[i]) for 0 <= i < K. The source symbol triples are determined using
the Triple generator defined in Section 5.4.4.4 as:

   For each i, 0 <= i < K

      (d[i], a[i], b[i]) = Trip[K, i]

5.4.2.3. Pre-Coding Relationships

The pre-coding relationships amongst the L intermediate symbols are
defined by expressing the last L-K intermediate symbols in terms of
the first K intermediate symbols.

The last L-K intermediate symbols C[K],...,C[L-1] consist of S LDPC
symbols and H Half symbols The values of S and H are determined from
K as described below. Then L = K+S+H.

Let

    X be the smallest positive integer such that X*(X-1) >= 2*K.

    S be the smallest prime integer such that S >= ceil(0.01*K) + X

    H be the smallest integer such that choose(H,ceil(H/2)) >= K + S

    H' = ceil(H/2)

    L = K+S+H

    C[0],...,C[K-1] denote the first K intermediate symbols

    C[K],...,C[K+S-1] denote the S LDPC symbols, initialised to zero

    C[K+S],...,C[L-1] denote the H Half symbols, initialised to zero

The S LDPC symbols are defined to be the values of C[K],...,C[K+S-1] at the end of the following process:

    For i = 0,...,K-1 do

       a = 1 + (floor(i/S) % (S-1))

       b = i % S

       C[K + b] = C[K + b] ^ C[i]

       b = (b + a) % S

       C[K + b] = C[K + b] ^ C[i]

       b = (b + a) % S

       C[K + b] = C[K + b] ^ C[i]

The H Half symbols are defined as follows:

Let

    g[i] = i ^ (floor(i/2)) for all positive integers i

       Note: g[i] is the Gray sequence, in which each element differs from the previous one in a single bit position

    m[k] denote the subsequence of g[.] whose elements have exactly k non-zero bits in their binary representation.

m[j,k] denote the jth element of the sequence m[k], where j=0, 1,
2, ...

Then, the Half symbols are defined as the values of C[K+S],...,C[L-1]
after the following process:

For h = 0,...,H-1 do

   For j = 0,...,K+S-1 do

      If bit h of m[j,H'] is equal to 1 then C[h+K+S] = C[h+K+S] ^
      C[j].

5.4.2.4. Intermediate Symbols

5.4.2.4.1. Definition

Given the K source symbols C'[0], C'[1],..., C'[K-1] the L
intermediate symbols C[0], C[1],..., C[L-1] are the uniquely defined
symbol values that satisfy the following conditions:

   1. The K source symbols C'[0], C'[1],..., C'[K-1] satisfy the K
   constraints

      C'[i] = LTEnc[K, (C[0],..., C[L-1]), (d[i], a[i], b[i])], for
      all i, 0 <= i < K.

   2. The L intermediate symbols C[0], C[1],..., C[L-1] satisfy the
   pre-coding relationships defined in Section 5.4.2.3.

5.4.2.4.2. Example Method for Calculation of Intermediate Symbols

This subsection describes a possible method for calculation of the L
intermediate symbols C[0], C[1],..., C[L-1] satisfying the
constraints in Section 5.4.2.4.1.

The 'generator matrix' for a code that generates N output symbols
from K input symbols is an NxK matrix over GF(2), where each row
corresponds to one of the output symbols and each column to one of
the input symbols and where the ith output symbol is equal to the sum
of those input symbols whose column contains a non-zero entry in row
i.

Then, the L intermediate symbols can be calculated as follows:

Let

C denote the column vector of the L intermediate symbols, C[0], C[1],..., C[L-1].

D denote the column vector consisting of S+H zero symbols followed by the K source symbols C'[0], C'[1], ..., C'[K-1]

Then the above constraints define an LxL matrix over GF(2), A, such that:

A*C = D

The matrix A can be constructed as follows:

Let:

G_LDPC be the S x K generator matrix of the LDPC symbols. So,

G_LDPC * Transpose[(C[0],...., C[K-1])] = Transpose[(C[K], ..., C[K+S-1])]

G_Half be the H x (K+S) generator matrix of the Half symbols, So,

G_Half * Transpose[(C[0], ..., C[S+K-1])] = Transpose[(C[K+S], ..., C[K+S+H-1])]

I_S be the S x S identity matrix

I_H be the H x H identity matrix

0_SxH be the S x H zero matrix

G_LT be the KxL generator matrix of the encoding symbols generated by the LT Encoder. So,

G_LT * Transpose[(C[0], ..., C[L-1])] = Transpose[(C'[0],C'[1],...,C'[K-1])]

i.e., G_LT(i,j) = 1 if and only if C[j] is included in the symbols that are XORed to produce LTEnc[K, (C[0], ..., C[L-1]), (d[i], a[i], b[i])].

Then:

The first S rows of A are equal to G_LDPC | I_S | 0_SxH.

The next H rows of A are equal to G_Half | I_H.

The remaining K rows of A are equal to G_LT.

The matrix A is depicted in Figure 4 below:

```
                    K                  S        H
      +------------------------+-------+-------+
      |                        |       |       |
      |                        |       |       |
   S  |         G_LDPC         |  I_S  | 0_SxH |
      |                        |       |       |
      |                        |       |       |
      +------------------------+-------+-------+
      |                        |       |       |
   H  |         G_Half         |       |  I_H  |
      |                        |       |       |
      +------------------------+-------+-------+
      |                                        |
      |                                        |
      |                                        |
   K  |                  G_LT                  |
      |                                        |
      |                                        |
      |                                        |
      +----------------------------------------+
```

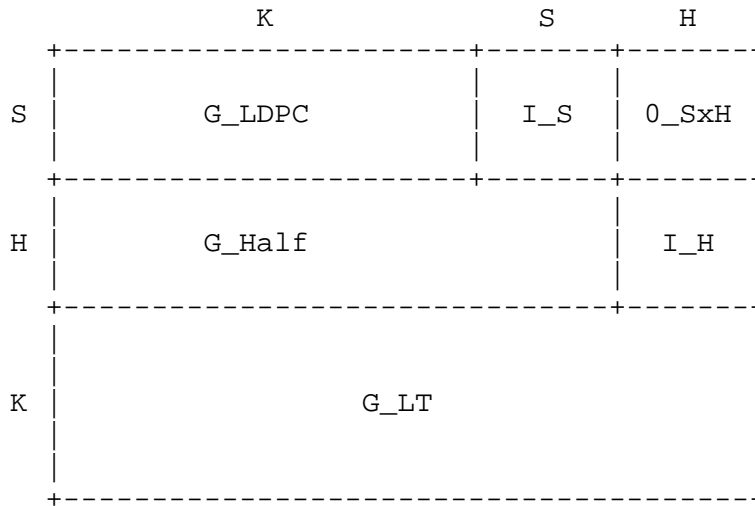Figure 4: The matrix A

The intermediate symbols can then be calculated as:

   C = (A^^-1)*D

The source symbol triples are generated such that for any K matrix, A
has full rank and is therefore invertible. This calculation can be
realized by applying a Raptor decoding process to the K source
symbols C'[0], C'[1],..., C'[K-1] to produce the L intermediate
symbols C[0], C[1],..., C[L-1].

To efficiently generate the intermediate symbols from the source
symbols, it is recommended that an efficient decoder implementation
such as that described in Section 5.5 be used. The source symbol
triples are designed to facilitate efficient decoding of the source
symbols using that algorithm.

5.4.3. Second Encoding Step: LT Encoding

In the second encoding step, the repair symbol with ESI X is
generated by applying the generator LTEnc[K, (C[0], C[1],...,
C[L-1]), (d, a, b)] defined in Section 5.4.4.3 to the L intermediate
symbols C[0], C[1],..., C[L-1] using the triple (d, a, b)=Trip[K,X]
generated according to Section 5.3.2

5.4.4. Generators

5.4.4.1. Random Generator

   The random number generator Rand[X, i, m] is defined as follows,
   where X is a non-negative integer, i is a non-negative integer, and m
   is a positive integer and the value produced is an integer between 0
   and m-1. Let V0 and V1 be arrays of 256 entries each, where each
   entry is a 4-byte unsigned integer. These arrays are provided in
   Section 5.6.

   Then,

      Rand[X, i, m] = (V0[(X + i) % 256] ^ V1[(floor(X/256)+ i) % 256])
      % m

5.4.4.2. Degree Generator

   The degree generator Deg[v] is defined as follows, where v is an
   integer that is at least 0 and less than 2^^20 = 1048576.

   In Table 1, find the index j such that f[j-1] <= v < f[j]

   Then, Deg[v] = d[j]

Index j	f[j]	d[j]
0	0	--
1	10241	1
2	491582	2
3	712794	3
4	831695	4
5	948446	10
6	1032189	11
7	1048576	40

        Table 1: Defines the degree distribution for encoding symbols

5.4.4.3. LT Encoding Symbol Generator

   The encoding symbol generator LTEnc[K, (C[0], C[1],..., C[L-1]), (d,
   a, b)] takes the following inputs:

K is the number of source symbols (or sub-symbols) for the source
block (sub-block). Let L be derived from K as described in
Section 5.4.2.3, and let L' be the smallest prime integer greater
than or equal to L.

(C[0], C[1],..., C[L-1]) is the array of L intermediate symbols
(sub-symbols) generated as described in Section 5.4.2.4.

(d, a, b) is a source triple determined using the Triple generator
defined in Section 5.4.4.4, whereby

   d is an integer denoting an encoding symbol degree

   a is an integer between 1 and L'-1 inclusive

   b is an integer between 0 and L'-1 inclusive

The encoding symbol generator produces a single encoding symbol as
output, according to the following algorithm:

   While (b >= L) do b = (b + a) % L'

   Let result = C[b].

   For j = 1,...,min(d-1,L-1) do

      b = (b + a) % L'

      While (b >= L) do b = (b + a) % L'

      result = result ^ C[b]

   Return result

5.4.4.4. Triple Generator

The triple generator Trip[K,X] takes the following inputs:

   K - The number of source symbols

   X - An encoding symbol ID

Let

   L be determined from K as described in Section 5.4.2.3

   L' be the smallest prime that is greater than or equal to L

Q = 65521, the largest prime smaller than 2^^16.

J(K) be the systematic index associated with K, as defined in
Section 5.7.

The output of the triple generator is a triple, (d, a, b) determined
as follows:

A = (53591 + J(K)*997) % Q

B = 10267*(J(K)+1) % Q

Y = (B + X*A) % Q

v = Rand[Y, 0, 2^^20]

d = Deg[v]

a = 1 + Rand[Y, 1, L'-1]

b = Rand[Y, 2, L']

5.5. Example FEC Decoder

5.5.1. General

This section describes an efficient decoding algorithm for the Raptor
codes described in this specification. Note that each received
encoding symbol can be considered as the value of an equation amongst
the intermediate symbols. From these simultaneous equations, and the
known pre-coding relationships amongst the intermediate symbols, any
algorithm for solving simultaneous equations can successfully decode
the intermediate symbols and hence the source symbols. However, the
algorithm chosen has a major effect on the computational efficiency
of the decoding.

5.5.2. Decoding a Source Block

5.5.2.1. General

It is assumed that the decoder knows the structure of the source
block it is to decode, including the symbol size, T, and the number K
of symbols in the source block.

From the algorithms described in Section 5.4, the Raptor decoder can
calculate the total number L = K+S+H of pre-coding symbols and
determine how they were generated from the source block to be
decoded. In this description, it is assumed that the received

encoding symbols for the source block to be decoded are passed to the
decoder. Note that, as described in Section 5.3.2, the last source
symbol of a source packet may have included padding bytes added for
FEC encoding purposes. These padding bytes may not be actually
included in the packet sent and so must be reinserted at the received
before passing the symbol to the decoder.

For each such encoding symbol, it is assumed that the number and set
of intermediate symbols whose exclusive-or is equal to the encoding
symbol is also passed to the decoder. In the case of source symbols,
the source symbol triples described in Section 5.4.2.2 indicate the
number and set of intermediate symbols that sum to give each source
symbol.

Let N >= K be the number of received encoding symbols for a source
block and let M = S+H+N. The following M by L bit matrix A can be
derived from the information passed to the decoder for the source
block to be decoded. Let C be the column vector of the L
intermediate symbols, and let D be the column vector of M symbols
with values known to the receiver, where the first S+H of the M
symbols are zero-valued symbols that correspond to LDPC and Half
symbols (these are check symbols for the LDPC and Half symbols, and
not the LDPC and Half symbols themselves), and the remaining N of the
M symbols are the received encoding symbols for the source block.
Then, A is the bit matrix that satisfies A*C = D, where here *
denotes matrix multiplication over GF[2]. In particular, A[i,j] = 1
if the intermediate symbol corresponding to index j is exclusive-ORed
into the LDPC, Half, or encoding symbol corresponding to index i in
the encoding, or if index i corresponds to a LDPC or Half symbol and
index j corresponds to the same LDPC or Half symbol. For all other i
and j, A[i,j] = 0.

Decoding a source block is equivalent to decoding C from known A and
D. It is clear that C can be decoded if and only if the rank of A
over GF[2] is L. Once C has been decoded, missing source symbols can
be obtained by using the source symbol triples to determine the
number and set of intermediate symbols that MUST be exclusive-ORed to
obtain each missing source symbol.

The first step in decoding C is to form a decoding schedule. In this
step A is converted, using Gaussian elimination (using row operations
and row and column reorderings) and after discarding M - L rows, into
the L by L identity matrix. The decoding schedule consists of the
sequence of row operations and row and column reorderings during the
Gaussian elimination process, and only depends on A and not on D.
 The decoding of C from D can take place concurrently with the
forming of the decoding schedule, or the decoding can take place
afterwards based on the decoding schedule.

The correspondence between the decoding schedule and the decoding of
C is as follows. Let c[0] = 0, c[1] = 1,...,c[L-1] = L-1 and d[0] =
0, d[1] = 1,...,d[M-1] = M-1 initially.

-   Each time row i of A is exclusive-ORed into row i' in the decoding
    schedule, then in the decoding process, symbol D[d[i]] is
    exclusive-ORed into symbol D[d[i']].

-   Each time row i is exchanged with row i' in the decoding schedule,
    then in the decoding process, the value of d[i] is exchanged with
    the value of d[i'].

-   Each time column j is exchanged with column j' in the decoding
    schedule, then in the decoding process, the value of c[j] is
    exchanged with the value of c[j'].

From this correspondence, it is clear that the total number of
exclusive-ORs of symbols in the decoding of the source block is the
number of row operations (not exchanges) in the Gaussian elimination.
Since A is the L by L identity matrix after the Gaussian elimination
and after discarding the last M - L rows, it is clear at the end of
successful decoding that the L symbols D[d[0]], D[d[1]],...,
D[d[L-1]] are the values of the L symbols C[c[0]], C[c[1]],...,
C[c[L-1]].

The order in which Gaussian elimination is performed to form the
decoding schedule has no bearing on whether or not the decoding is
successful. However, the speed of the decoding depends heavily on
the order in which Gaussian elimination is performed. (Furthermore,
maintaining a sparse representation of A is crucial, although this is
not described here). The remainder of this section describes an
order in which Gaussian elimination could be performed that is
relatively efficient.

5.5.2.2. First Phase

The first phase of the Gaussian elimination, the matrix A, is
conceptually partitioned into submatrices. The submatrix sizes are
parameterized by non-negative integers i and u, which are initialized
to 0. The submatrices of A are:

(1) The submatrix I defined by the intersection of the first i
    rows and first i columns. This is the identity matrix at the
    end of each step in the phase.

(2) The submatrix defined by the intersection of the first i rows
    and all but the first i columns and last u columns. All
    entries of this submatrix are zero.

   (3) The submatrix defined by the intersection of the first i
       columns and all but the first i rows. All entries of this
       submatrix are zero.

   (4) The submatrix U defined by the intersection of all the rows
       and the last u columns.

   (5) The submatrix V formed by the intersection of all but the
       first i columns and the last u columns and all but the first i
       rows.

Figure 5 illustrates the submatrices of A. At the beginning of the
first phase, V = A. In each step, a row of A is chosen.

```
+----------+----------------+---------+
|          |                |         |
|    I     |   All Zeros    |         |
|          |                |         |
+----------+----------------+    U    |
|          |                |         |
|          |                |         |
| All Zeros|       V        |         |
|          |                |         |
|          |                |         |
+----------+----------------+---------+
```
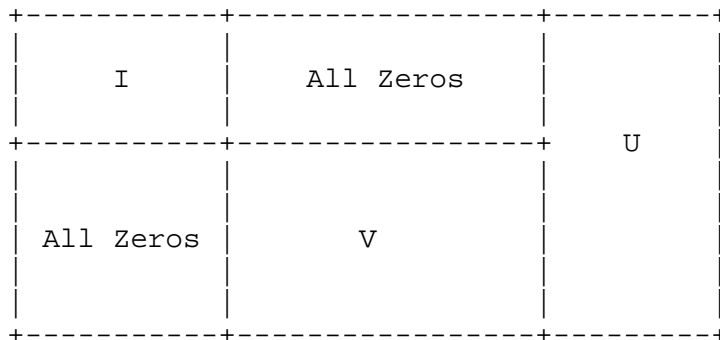
              Figure 5: Submatrices of A in the first phase

The following graph defined by the structure of V is used in
determining which row of A is chosen. The columns that intersect V
are the nodes in the graph, and the rows that have exactly 2 ones in
V are the edges of the graph that connect the two columns (nodes) in
the positions of the two ones. A component in this graph is a
maximal set of nodes (columns) and edges (rows) such that there is a
path between each pair of nodes/edges in the graph. The size of a
component is the number of nodes (columns) in the component.

There are at most L steps in the first phase. The phase ends
successfully when i + u = L, i.e., when V and the all-zeroes
submatrix above V have disappeared and A consists of I, the all
zeroes submatrix below I, and U. The phase ends unsuccessfully in
decoding failure if, at some step before V disappears, there is no
non-zero row in V to choose in that step. Whenever there are non-
zero rows in V, then the next step starts by choosing a row of A as
follows:

   o Let r be the minimum integer such that at least one row of A has
      exactly r ones in V.

      * If r != 2, then choose a row with exactly r ones in V with
         minimum original degree among all such rows.

      * If r = 2, then choose any row with exactly 2 ones in V that is
         part of a maximum size component in the graph defined by V.

   After the row is chosen in this step the first row of A that
   intersects V is exchanged with the chosen row so that the chosen row
   is the first row that intersects V. The columns of A among those
   that intersect V are reordered so that one of the r ones in the
   chosen row appears in the first column of V and so that the remaining
   r-1 ones appear in the last columns of V. Then, the chosen row is
   exclusive-ORed into all the other rows of A below the chosen row that
   have a one in the first column of V. Finally, i is incremented by 1
   and u is incremented by r-1, which completes the step.

5.5.2.3. Second Phase

   The submatrix U is further partitioned into the first i rows,
   U_upper, and the remaining M - i rows, U_lower. Gaussian elimination
   is performed in the second phase on U_lower to either determine that
   its rank is less than u (decoding failure) or to convert it into a
   matrix where the first u rows is the identity matrix (success of the
   second phase). Call this u by u identity matrix I_u. The M - L rows
   of A that intersect U_lower - I_u are discarded. After this phase, A
   has L rows and L columns.

5.5.2.4. Third Phase

   After the second phase, the only portion of A that needs to be zeroed
   out to finish converting A into the L by L identity matrix is
   U_upper. The number of rows i of the submatrix U_upper is generally
   much larger than the number of columns u of U_upper. To zero out
   U_upper efficiently, the following precomputation matrix U' is
   computed based on I_u in the third phase and then U' is used in the
   fourth phase to zero out U_upper. The u rows of Iu are partitioned
   into ceil(u/8) groups of 8 rows each. Then, for each group of 8
   rows, all non-zero combinations of the 8 rows are computed, resulting
   in 2^^8 - 1 = 255 rows (this can be done with 2^^8-8-1 = 247
   exclusive-ors of rows per group, since the combinations of Hamming
   weight one that appear in I_u do not need to be recomputed). Thus,
   the resulting precomputation matrix U' has ceil(u/8)*255 rows and u
   columns. Note that U' is not formally a part of matrix A, but will
   be used in the fourth phase to zero out U_upper.

5.5.2.5. Fourth Phase

   For each of the first i rows of A, for each group of 8 columns in the
   U_upper submatrix of this row, if the set of 8 column entries in
   U_upper are not all zero, then the row of the precomputation matrix
   U' that matches the pattern in the 8 columns is exclusive-ORed into
   the row, thus zeroing out those 8 columns in the row at the cost of
   exclusive-ORing one row of U' into the row.

   After this phase, A is the L by L identity matrix and a complete
   decoding schedule has been successfully formed. Then, as explained
   in Section 5.5.2.1, the corresponding decoding consisting of
   exclusive-ORing known encoding symbols can be executed to recover the
   intermediate symbols based on the decoding schedule. The triples
   associated with all source symbols are computed according to
   Section 5.4.2.2. The triples for received source symbols are used in
   the decoding. The triples for missing source symbols are used to
   determine which intermediate symbols need to be exclusive-ORed to
   recover the missing source symbols.

5.6. Random Numbers

   The two tables V0 and V1 described in Section 5.4.4.1 are given
   below. Each entry is a 32-bit integer in decimal representation.

5.6.1. The Table V0

   251291136, 3952231631, 3370958628, 4070167936, 123631495, 3351110283,
   3218676425, 2011642291, 774603218, 2402805061, 1004366930,
   1843948209, 428891132, 3746331984, 1591258008, 3067016507,
   1433388735, 504005498, 2032657933, 3419319784, 2805686246,
   3102436986, 3808671154, 2501582075, 3978944421, 246043949,
   4016898363, 649743608, 1974987508, 2651273766, 2357956801, 689605112,
   715807172, 2722736134, 191939188, 3535520147, 3277019569, 1470435941,
   3763101702, 3232409631, 122701163, 3920852693, 782246947, 372121310,
   2995604341, 2045698575, 2332962102, 4005368743, 218596347,
   3415381967, 4207612806, 861117671, 3676575285, 2581671944,
   3312220480, 681232419, 307306866, 4112503940, 1158111502, 709227802,
   2724140433, 4201101115, 4215970289, 4048876515, 3031661061,
   1909085522, 510985033, 1361682810, 129243379, 3142379587, 2569842483,
   3033268270, 1658118006, 932109358, 1982290045, 2983082771,
   3007670818, 3448104768, 683749698, 778296777, 1399125101, 1939403708,
   1692176003, 3868299200, 1422476658, 593093658, 1878973865,
   2526292949, 1591602827, 3986158854, 3964389521, 2695031039,
   1942050155, 424618399, 1347204291, 2669179716, 2434425874,
   2540801947, 1384069776, 4123580443, 1523670218, 2708475297,
   1046771089, 2229796016, 1255426612, 4213663089, 1521339547,
   3041843489, 420130494, 10677091, 515623176, 3457502702, 2115821274,

      2720124766, 3242576090, 854310108, 425973987, 325832382, 1796851292,
      2462744411, 1976681690, 1408671665, 1228817808, 3917210003,
      263976645, 2593736473, 2471651269, 4291353919, 650792940, 1191583883,
      3046561335, 2466530435, 2545983082, 969168436, 2019348792,
      2268075521, 1169345068, 3250240009, 3963499681, 2560755113,
      911182396, 760842409, 3569308693, 2687243553, 381854665, 2613828404,
      2761078866, 1456668111, 883760091, 3294951678, 1604598575,
      1985308198, 1014570543, 2724959607, 3062518035, 3115293053,
      138853680, 4160398285, 3322241130, 2068983570, 2247491078,
      3669524410, 1575146607, 828029864, 3732001371, 3422026452,
      3370954177, 4006626915, 543812220, 1243116171, 3928372514,
      2791443445, 4081325272, 2280435605, 885616073, 616452097, 3188863436,
      2780382310, 2340014831, 1208439576, 258356309, 3837963200,
      2075009450, 3214181212, 3303882142, 880813252, 1355575717, 207231484,
      2420803184, 358923368, 1617557768, 3272161958, 1771154147,
      2842106362, 1751209208, 1421030790, 658316681, 194065839, 3241510581,
      38625260, 301875395, 4176141739, 297312930, 2137802113, 1502984205,
      3669376622, 3728477036, 234652930, 2213589897, 2734638932,
      1129721478, 3187422815, 2859178611, 3284308411, 3819792700,
      3557526733, 451874476, 1740576081, 3592838701, 1709429513,
      3702918379, 3533351328, 1641660745, 179350258, 2380520112,
      3936163904, 3685256204, 3156252216, 1854258901, 2861641019,
      3176611298, 834787554, 331353807, 517858103, 3010168884, 4012642001,
      2217188075, 3756943137, 3077882590, 2054995199, 3081443129,
      3895398812, 1141097543, 2376261053, 2626898255, 2554703076,
      401233789, 1460049922, 678083952, 1064990737, 940909784, 1673396780,
      528881783, 1712547446, 3629685652, 1358307511

5.6.2. The Table V1

      807385413, 2043073223, 3336749796, 1302105833, 2278607931, 541015020,
      1684564270, 372709334, 3508252125, 1768346005, 1270451292,
      2603029534, 2049387273, 3891424859, 2152948345, 4114760273,
      915180310, 3754787998, 700503826, 2131559305, 1308908630, 224437350,
      4065424007, 3638665944, 1679385496, 3431345226, 1779595665,
      3068494238, 1424062773, 1033448464, 4050396853, 3302235057,
      420600373, 2868446243, 311689386, 259047959, 4057180909, 1575367248,
      4151214153, 110249784, 3006865921, 4293710613, 3501256572, 998007483,
      499288295, 1205710710, 2997199489, 640417429, 3044194711, 486690751,
      2686640734, 2394526209, 2521660077, 49993987, 3843885867, 4201106668,
      415906198, 19296841, 2402488407, 2137119134, 1744097284, 579965637,
      2037662632, 852173610, 2681403713, 1047144830, 2982173936, 910285038,
      4187576520, 2589870048, 989448887, 3292758024, 506322719, 176010738,
      1865471968, 2619324712, 564829442, 1996870325, 339697593, 4071072948,
      3618966336, 2111320126, 1093955153, 957978696, 892010560, 1854601078,
      1873407527, 2498544695, 2694156259, 1927339682, 1650555729,
      183933047, 3061444337, 2067387204, 228962564, 3904109414, 1595995433,
      1780701372, 2463145963, 307281463, 3237929991, 3852995239,

```
2398693510, 3754138664, 522074127, 146352474, 4104915256, 3029415884,
3545667983, 332038910, 976628269, 3123492423, 3041418372, 2258059298,
2139377204, 3243642973, 3226247917, 3674004636, 2698992189,
3453843574, 1963216666, 3509855005, 2358481858, 747331248,
1957348676, 1097574450, 2435697214, 3870972145, 1888833893,
2914085525, 4161315584, 1273113343, 3269644828, 3681293816,
412536684, 1156034077, 3823026442, 1066971017, 3598330293,
1979273937, 2079029895, 1195045909, 1071986421, 2712821515,
3377754595, 2184151095, 750918864, 2585729879, 4249895712,
1832579367, 1192240192, 946734366, 31230688, 3174399083, 3549375728,
1642430184, 1904857554, 861877404, 3277825584, 4267074718,
3122860549, 666423581, 644189126, 226475395, 307789415, 1196105631,
3191691839, 782852669, 1608507813, 1847685900, 4069766876,
3931548641, 2526471011, 766865139, 2115084288, 4259411376,
3323683436, 568512177, 3736601419, 1800276898, 4012458395, 1823982,
27980198, 2023839966, 869505096, 431161506, 1024804023, 1853869307,
3393537983, 1500703614, 3019471560, 1351086955, 3096933631,
3034634988, 2544598006, 1230942551, 3362230798, 159984793, 491590373,
3993872886, 3681855622, 903593547, 3535062472, 1799803217, 772984149,
895863112, 1899036275, 4187322100, 101856048, 234650315, 3183125617,
3190039692, 525584357, 1286834489, 455810374, 1869181575, 922673938,
3877430102, 3422391938, 1414347295, 1971054608, 3061798054,
830555096, 2822905141, 167033190, 1079139428, 4210126723, 3593797804,
429192890, 372093950, 1779187770, 3312189287, 204349348, 452421568,
2800540462, 3733109044, 1235082423, 1765319556, 3174729780,
3762994475, 3171962488, 442160826, 198349622, 45942637, 1324086311,
2901868599, 678860040, 3812229107, 19936821, 1119590141, 3640121682,
3545931032, 2102949142, 2828208598, 3603378023, 4135048896
```

5.7. Systematic Indices J(K)

   For each value of K, the systematic index J(K) is designed to have
   the property that the set of source symbol triples (d[0], a[0],
   b[0]), ..., (d[L-1], a[L-1], b[L-1]) are such that the L intermediate
   symbols are uniquely defined, i.e., the matrix A in Section 5.4.2.4.2
   has full rank and is therefore invertible.

   The following is the list of the systematic indices for values of K
   between 4 and 8192 inclusive.

```
18, 14, 61, 46, 14, 22, 20, 40, 48, 1, 29, 40, 43, 46, 18, 8, 20, 2,
61, 26, 13, 29, 36, 19, 58, 5, 58, 0, 54, 56, 24, 14, 5, 67, 39, 31,
25, 29, 24, 19, 14, 56, 49, 49, 63, 30, 4, 39, 2, 1, 20, 19, 61, 4,
54, 70, 25, 52, 9, 26, 55, 69, 27, 68, 75, 19, 64, 57, 45, 3, 37, 31,
100, 41, 25, 41, 53, 23, 9, 31, 26, 30, 30, 46, 90, 50, 13, 90, 77,
61, 31, 54, 54, 3, 21, 66, 21, 11, 23, 11, 29, 21, 7, 1, 27, 4, 34,
17, 85, 69, 17, 75, 93, 57, 0, 53, 71, 88, 119, 88, 90, 22, 0, 58,
41, 22, 96, 26, 79, 118, 19, 3, 81, 72, 50, 0, 32, 79, 28, 25, 12,
```

```
25, 29, 3, 37, 30, 30, 41, 84, 32, 31, 61, 32, 61, 7, 56, 54, 39, 33,
66, 29, 3, 14, 75, 75, 78, 84, 75, 84, 25, 54, 25, 25, 107, 78, 27,
73, 0, 49, 96, 53, 50, 21, 10, 73, 58, 65, 27, 3, 27, 18, 54, 45, 69,
29, 3, 65, 31, 71, 76, 56, 54, 76, 54, 13, 5, 18, 142, 17, 3, 37,
114, 41, 25, 56, 0, 23, 3, 41, 22, 22, 31, 18, 48, 31, 58, 37, 75,
88, 3, 56, 1, 95, 19, 73, 52, 52, 4, 75, 26, 1, 25, 10, 1, 70, 31,
31, 12, 10, 54, 46, 11, 74, 84, 74, 8, 58, 23, 74, 8, 36, 11, 16, 94,
76, 14, 57, 65, 8, 22, 10, 36, 36, 96, 62, 103, 6, 75, 103, 58, 10,
15, 41, 75, 125, 58, 15, 10, 34, 29, 34, 4, 16, 29, 18, 18, 28, 71,
28, 43, 77, 18, 41, 41, 41, 62, 29, 96, 15, 106, 43, 15, 3, 43, 61,
3, 18, 103, 77, 29, 103, 19, 58, 84, 58, 1, 146, 32, 3, 70, 52, 54,
29, 70, 69, 124, 62, 1, 26, 38, 26, 3, 16, 26, 5, 51, 120, 41, 16, 1,
43, 34, 34, 29, 37, 56, 29, 96, 86, 54, 25, 84, 50, 34, 34, 93, 84,
96, 29, 29, 50, 50, 6, 1, 105, 78, 15, 37, 19, 50, 71, 36, 6, 54, 8,
28, 54, 75, 75, 16, 75, 131, 5, 25, 16, 69, 17, 69, 6, 96, 53, 96,
41, 119, 6, 6, 88, 50, 88, 52, 37, 0, 124, 73, 73, 7, 14, 36, 69, 79,
6, 114, 40, 79, 17, 77, 24, 44, 37, 69, 27, 37, 29, 33, 37, 50, 31,
69, 29, 101, 7, 61, 45, 17, 73, 37, 34, 18, 94, 22, 22, 63, 3, 25,
25, 17, 3, 90, 34, 34, 41, 34, 41, 54, 41, 54, 41, 41, 41, 163, 143,
96, 18, 32, 39, 86, 104, 11, 17, 17, 11, 86, 104, 78, 70, 52, 78, 17,
73, 91, 62, 7, 128, 50, 124, 18, 101, 46, 10, 75, 104, 73, 58, 132,
34, 13, 4, 95, 88, 33, 76, 74, 54, 62, 113, 114, 103, 32, 103, 69,
54, 53, 3, 11, 72, 31, 53, 102, 37, 53, 11, 81, 41, 10, 164, 10, 41,
31, 36, 113, 82, 3, 125, 62, 16, 4, 41, 41, 4, 128, 49, 138, 128, 74,
103, 0, 6, 101, 41, 142, 171, 39, 105, 121, 81, 62, 41, 81, 37, 3,
81, 69, 62, 3, 69, 70, 21, 29, 4, 91, 87, 37, 79, 36, 21, 71, 37, 41,
75, 128, 128, 15, 25, 3, 108, 73, 91, 62, 114, 62, 62, 36, 36, 15,
58, 114, 61, 114, 58, 105, 114, 41, 61, 176, 145, 46, 37, 30, 220,
77, 138, 15, 1, 128, 53, 50, 50, 58, 8, 91, 114, 105, 63, 91, 37, 37,
13, 169, 51, 102, 6, 102, 23, 105, 23, 58, 6, 29, 29, 19, 82, 29, 13,
36, 27, 29, 61, 12, 18, 127, 127, 12, 44, 102, 18, 4, 15, 206, 53,
127, 53, 17, 69, 69, 69, 29, 29, 109, 25, 102, 25, 53, 62, 99, 62,
62, 29, 62, 62, 45, 91, 125, 29, 29, 29, 4, 117, 72, 4, 30, 71, 71,
95, 79, 179, 71, 30, 53, 32, 32, 49, 25, 91, 25, 26, 26, 103, 123,
26, 41, 162, 78, 52, 103, 25, 6, 142, 94, 45, 45, 94, 127, 94, 94,
94, 47, 209, 138, 39, 39, 19, 154, 73, 67, 91, 27, 91, 84, 4, 84, 91,
12, 14, 165, 142, 54, 69, 192, 157, 185, 8, 95, 25, 62, 103, 103, 95,
71, 97, 62, 128, 0, 29, 51, 16, 94, 16, 16, 51, 0, 29, 85, 10, 105,
16, 29, 29, 13, 29, 4, 4, 132, 23, 95, 25, 54, 41, 29, 50, 70, 58,
142, 72, 70, 15, 72, 54, 29, 22, 145, 29, 127, 29, 85, 58, 101, 34,
165, 91, 46, 46, 25, 185, 25, 77, 128, 46, 128, 46, 188, 114, 46, 25,
45, 45, 114, 145, 114, 15, 102, 142, 8, 73, 31, 139, 157, 13, 79, 13,
114, 150, 8, 90, 91, 123, 69, 82, 132, 8, 18, 10, 102, 103, 114, 103,
8, 103, 13, 115, 55, 62, 3, 8, 154, 114, 99, 19, 8, 31, 73, 19, 99,
10, 6, 121, 32, 13, 32, 119, 32, 29, 145, 30, 13, 13, 114, 145, 32,
1, 123, 39, 29, 31, 69, 31, 140, 72, 72, 25, 25, 123, 25, 123, 8, 4,
85, 8, 25, 39, 25, 39, 85, 138, 25, 138, 25, 33, 102, 70, 25, 25, 31,
25, 25, 192, 69, 69, 114, 145, 120, 120, 8, 33, 98, 15, 212, 155, 8,
```

```
101, 8, 8, 98, 68, 155, 102, 132, 120, 30, 25, 123, 123, 101, 25,
123, 32, 24, 94, 145, 32, 24, 94, 118, 145, 101, 53, 53, 25, 128,
173, 142, 81, 81, 69, 33, 33, 125, 4, 1, 17, 27, 4, 17, 102, 27, 13,
25, 128, 71, 13, 39, 53, 13, 53, 47, 39, 23, 128, 53, 39, 47, 39,
135, 158, 136, 36, 36, 27, 157, 47, 76, 213, 47, 156, 25, 25, 53, 25,
53, 25, 86, 27, 159, 25, 62, 79, 39, 79, 25, 145, 49, 25, 143, 13,
114, 150, 130, 94, 102, 39, 4, 39, 61, 77, 228, 22, 25, 47, 119, 205,
122, 119, 205, 119, 22, 119, 258, 143, 22, 81, 179, 22, 22, 143, 25,
65, 53, 168, 36, 79, 175, 37, 79, 70, 79, 103, 70, 25, 175, 4, 96,
96, 49, 128, 138, 96, 22, 62, 47, 95, 105, 95, 62, 95, 62, 142, 103,
69, 103, 30, 103, 34, 173, 127, 70, 127, 132, 18, 85, 22, 71, 18,
206, 206, 18, 128, 145, 70, 193, 188, 8, 125, 114, 70, 128, 114, 145,
102, 25, 12, 108, 102, 94, 10, 102, 1, 102, 124, 22, 22, 118, 132,
22, 116, 75, 41, 63, 41, 189, 208, 55, 85, 69, 8, 71, 53, 71, 69,
102, 165, 41, 99, 69, 33, 33, 29, 156, 102, 13, 251, 102, 25, 13,
109, 102, 164, 102, 164, 102, 25, 29, 228, 29, 259, 179, 222, 95, 94,
30, 30, 30, 142, 55, 142, 72, 55, 102, 128, 17, 69, 164, 165, 3, 164,
36, 165, 27, 27, 45, 21, 21, 237, 113, 83, 231, 106, 13, 154, 13,
154, 128, 154, 148, 258, 25, 154, 128, 3, 27, 10, 145, 145, 21, 146,
25, 1, 185, 121, 0, 1, 95, 55, 95, 95, 30, 0, 27, 95, 0, 95, 8, 222,
27, 121, 30, 95, 121, 0, 98, 94, 131, 55, 95, 95, 30, 98, 30, 0, 91,
145, 66, 179, 66, 58, 175, 29, 0, 31, 173, 146, 160, 39, 53, 28, 123,
199, 123, 175, 146, 156, 54, 54, 149, 25, 70, 178, 128, 25, 70, 70,
94, 224, 54, 4, 54, 54, 25, 228, 160, 206, 165, 143, 206, 108, 220,
234, 160, 13, 169, 103, 103, 103, 91, 213, 222, 91, 103, 91, 103, 31,
30, 123, 13, 62, 103, 50, 106, 42, 13, 145, 114, 220, 65, 8, 8, 175,
11, 104, 94, 118, 132, 27, 118, 193, 27, 128, 127, 127, 183, 33, 30,
29, 103, 128, 61, 234, 165, 41, 29, 193, 33, 207, 41, 165, 165, 55,
81, 157, 157, 8, 81, 11, 27, 8, 8, 98, 96, 142, 145, 41, 179, 112,
62, 180, 206, 206, 165, 39, 241, 45, 151, 26, 197, 102, 192, 125,
128, 67, 128, 69, 128, 197, 33, 125, 102, 13, 103, 25, 30, 12, 30,
12, 30, 25, 77, 12, 25, 180, 27, 10, 69, 235, 228, 343, 118, 69, 41,
8, 69, 175, 25, 69, 25, 125, 41, 25, 41, 8, 155, 146, 155, 146, 155,
206, 168, 128, 157, 27, 273, 211, 211, 168, 11, 173, 154, 77, 173,
77, 102, 102, 102, 8, 85, 95, 102, 157, 28, 122, 234, 122, 157, 235,
222, 241, 10, 91, 179, 25, 13, 25, 41, 25, 206, 41, 6, 41, 158, 206,
206, 33, 296, 296, 33, 228, 69, 8, 114, 148, 33, 29, 66, 27, 27, 30,
233, 54, 173, 108, 106, 108, 108, 53, 103, 33, 33, 33, 176, 27, 27,
205, 164, 105, 237, 41, 27, 72, 165, 29, 29, 259, 132, 132, 132, 364,
71, 71, 27, 94, 160, 127, 51, 234, 55, 27, 95, 94, 165, 55, 55, 41,
0, 41, 128, 4, 123, 173, 6, 164, 157, 121, 121, 154, 86, 164, 164,
25, 93, 164, 25, 164, 210, 284, 62, 93, 30, 25, 25, 30, 30, 260, 130,
25, 125, 57, 53, 166, 166, 166, 185, 166, 158, 94, 113, 215, 159, 62,
99, 21, 172, 99, 184, 62, 259, 4, 21, 21, 77, 62, 173, 41, 146, 6,
41, 128, 121, 41, 11, 121, 103, 159, 164, 175, 206, 91, 103, 164, 72,
25, 129, 72, 206, 129, 33, 103, 102, 102, 29, 13, 11, 251, 234, 135,
31, 8, 123, 65, 91, 121, 129, 65, 243, 10, 91, 8, 65, 70, 228, 220,
243, 91, 10, 10, 30, 178, 91, 178, 33, 21, 25, 235, 165, 11, 161,
```

```
   158, 27, 27, 30, 128, 75, 36, 30, 36, 36, 173, 25, 33, 178, 112, 162,
   112, 112, 112, 162, 33, 33, 178, 123, 123, 39, 106, 91, 106, 106,
   158, 106, 106, 284, 39, 230, 21, 228, 11, 21, 228, 159, 241, 62, 10,
   62, 10, 68, 234, 39, 39, 138, 62, 22, 27, 183, 22, 215, 10, 175, 175,
   353, 228, 42, 193, 175, 175, 27, 98, 27, 193, 150, 27, 173, 17, 233,
   233, 25, 102, 123, 152, 242, 108, 4, 94, 176, 13, 41, 219, 17, 151,
   22, 103, 103, 53, 128, 233, 284, 25, 265, 128, 39, 39, 138, 42, 39,
   21, 86, 95, 127, 29, 91, 46, 103, 103, 215, 25, 123, 123, 230, 25,
   193, 180, 30, 60, 30, 242, 136, 180, 193, 30, 206, 180, 60, 165, 206,
   193, 165, 123, 164, 103, 68, 25, 70, 91, 25, 82, 53, 82, 186, 53, 82,
   53, 25, 30, 282, 91, 13, 234, 160, 160, 126, 149, 36, 36, 160, 149,
   178, 160, 39, 294, 149, 149, 160, 39, 95, 221, 186, 106, 178, 316,
   267, 53, 53, 164, 159, 164, 165, 94, 228, 53, 52, 178, 183, 53, 294,
   128, 55, 140, 294, 25, 95, 366, 15, 304, 13, 183, 77, 230, 6, 136,
   235, 121, 311, 273, 36, 158, 235, 230, 98, 201, 165, 165, 165, 91,
   175, 248, 39, 185, 128, 39, 39, 128, 313, 91, 36, 219, 130, 25, 130,
   234, 234, 130, 234, 121, 205, 304, 94, 77, 64, 259, 60, 60, 60, 77,
   242, 60, 145, 95, 270, 18, 91, 199, 159, 91, 235, 58, 249, 26, 123,
   114, 29, 15, 191, 15, 30, 55, 55, 347, 4, 29, 15, 4, 341, 93, 7, 30,
   23, 7, 121, 266, 178, 261, 70, 169, 25, 25, 158, 169, 25, 169, 270,
   270, 13, 128, 327, 103, 55, 128, 103, 136, 159, 103, 327, 41, 32,
   111, 111, 114, 173, 215, 173, 25, 173, 180, 114, 173, 173, 98, 93,
   25, 160, 157, 159, 160, 159, 159, 160, 320, 35, 193, 221, 33, 36,
   136, 248, 91, 215, 125, 215, 156, 68, 125, 125, 1, 287, 123, 94, 30,
   184, 13, 30, 94, 123, 206, 12, 206, 289, 128, 122, 184, 128, 289,
   178, 29, 26, 206, 178, 65, 206, 128, 192, 102, 197, 36, 94, 94, 155,
   10, 36, 121, 280, 121, 368, 192, 121, 121, 179, 121, 36, 54, 192,
   121, 192, 197, 118, 123, 224, 118, 10, 192, 10, 91, 269, 91, 49, 206,
   184, 185, 62, 8, 49, 289, 30, 5, 55, 30, 42, 39, 220, 298, 42, 347,
   42, 234, 42, 70, 42, 55, 321, 129, 172, 173, 172, 13, 98, 129, 325,
   235, 284, 362, 129, 233, 345, 175, 261, 175, 60, 261, 58, 289, 99,
   99, 99, 206, 99, 36, 175, 29, 25, 432, 125, 264, 168, 173, 69, 158,
   273, 179, 164, 69, 158, 69, 8, 95, 192, 30, 164, 101, 44, 53, 273,
   335, 273, 53, 45, 128, 45, 234, 123, 105, 103, 103, 224, 36, 90, 211,
   282, 264, 91, 228, 91, 166, 264, 228, 398, 50, 101, 91, 264, 73, 36,
   25, 73, 50, 50, 242, 36, 36, 58, 165, 204, 353, 165, 125, 320, 128,
   298, 298, 180, 128, 60, 102, 30, 30, 53, 179, 234, 325, 234, 175, 21,
   250, 215, 103, 21, 21, 250, 91, 211, 91, 313, 301, 323, 215, 228,
   160, 29, 29, 81, 53, 180, 146, 248, 66, 159, 39, 98, 323, 98, 36, 95,
   218, 234, 39, 82, 82, 230, 62, 13, 62, 230, 13, 30, 98, 0, 8, 98, 8,
   98, 91, 267, 121, 197, 30, 78, 27, 78, 102, 27, 298, 160, 103, 264,
   264, 264, 175, 17, 273, 273, 165, 31, 160, 17, 99, 17, 99, 234, 31,
   17, 99, 36, 26, 128, 29, 214, 353, 264, 102, 36, 102, 264, 264, 273,
   273, 4, 16, 138, 138, 264, 128, 313, 25, 420, 60, 10, 280, 264, 60,
   60, 103, 178, 125, 178, 29, 327, 29, 36, 30, 36, 4, 52, 183, 183,
   173, 52, 31, 173, 31, 158, 31, 158, 31, 9, 31, 31, 353, 31, 353, 173,
   415, 9, 17, 222, 31, 103, 31, 165, 27, 31, 31, 165, 27, 27, 206, 31,
   31, 4, 4, 30, 4, 4, 264, 185, 159, 310, 273, 310, 173, 40, 4, 173, 4,
```

```
173, 4, 250, 250, 62, 188, 119, 250, 233, 62, 121, 105, 105, 54, 103,
111, 291, 236, 236, 103, 297, 36, 26, 316, 69, 183, 158, 206, 129,
160, 129, 184, 55, 179, 279, 11, 179, 347, 160, 184, 129, 179, 351,
179, 353, 179, 129, 129, 351, 11, 111, 93, 93, 235, 103, 173, 53, 93,
50, 111, 86, 123, 94, 36, 183, 60, 55, 55, 178, 219, 253, 321, 178,
235, 235, 183, 183, 204, 321, 219, 160, 193, 335, 121, 70, 69, 295,
159, 297, 231, 121, 231, 136, 353, 136, 121, 279, 215, 366, 215, 353,
159, 353, 353, 103, 31, 31, 298, 298, 30, 30, 165, 273, 25, 219, 35,
165, 259, 54, 36, 54, 54, 165, 71, 250, 327, 13, 289, 165, 196, 165,
165, 94, 233, 165, 94, 60, 165, 96, 220, 166, 271, 158, 397, 122, 53,
53, 137, 280, 272, 62, 30, 30, 30, 105, 102, 67, 140, 8, 67, 21, 270,
298, 69, 173, 298, 91, 179, 327, 86, 179, 88, 179, 179, 55, 123, 220,
233, 94, 94, 175, 13, 53, 13, 154, 191, 74, 83, 83, 325, 207, 83, 74,
83, 325, 74, 316, 388, 55, 55, 364, 55, 183, 434, 273, 273, 273, 164,
213, 11, 213, 327, 321, 21, 352, 185, 103, 13, 13, 55, 30, 323, 123,
178, 435, 178, 30, 175, 175, 30, 481, 527, 175, 125, 232, 306, 232,
206, 306, 364, 206, 270, 206, 232, 10, 30, 130, 160, 130, 347, 240,
30, 136, 130, 347, 136, 279, 298, 206, 30, 103, 273, 241, 70, 206,
306, 434, 206, 94, 94, 156, 161, 321, 321, 64, 161, 13, 183, 183, 83,
161, 13, 169, 13, 159, 36, 173, 159, 36, 36, 230, 235, 235, 159, 159,
335, 312, 42, 342, 264, 39, 39, 39, 34, 298, 36, 36, 252, 164, 29,
493, 29, 387, 387, 435, 493, 132, 273, 105, 132, 74, 73, 206, 234,
273, 206, 95, 15, 280, 280, 280, 280, 397, 273, 273, 242, 397, 280,
397, 397, 397, 273, 397, 280, 230, 137, 353, 67, 81, 137, 137, 353,
259, 312, 114, 164, 164, 25, 77, 21, 77, 165, 30, 30, 231, 234, 121,
234, 312, 121, 364, 136, 123, 123, 136, 123, 136, 150, 264, 285, 30,
166, 93, 30, 39, 224, 136, 39, 355, 355, 397, 67, 67, 25, 67, 25,
298, 11, 67, 264, 374, 99, 150, 321, 67, 70, 67, 295, 150, 29, 321,
150, 70, 29, 142, 355, 311, 173, 13, 253, 103, 114, 114, 70, 192, 22,
128, 128, 183, 184, 70, 77, 215, 102, 292, 30, 123, 279, 292, 142,
33, 215, 102, 468, 123, 468, 473, 30, 292, 215, 30, 213, 443, 473,
215, 234, 279, 279, 279, 279, 265, 443, 206, 66, 313, 34, 30, 206,
30, 51, 15, 206, 41, 434, 41, 398, 67, 30, 301, 67, 36, 3, 285, 437,
136, 136, 22, 136, 145, 365, 323, 323, 145, 136, 22, 453, 99, 323,
353, 9, 258, 323, 231, 128, 231, 382, 150, 420, 39, 94, 29, 29, 353,
22, 22, 347, 353, 39, 29, 22, 183, 8, 284, 355, 388, 284, 60, 64, 99,
60, 64, 150, 95, 150, 364, 150, 95, 150, 6, 236, 383, 544, 81, 206,
388, 206, 58, 159, 99, 231, 228, 363, 363, 121, 99, 121, 121, 99,
422, 544, 273, 173, 121, 427, 102, 121, 235, 284, 179, 25, 197, 25,
179, 511, 70, 368, 70, 25, 388, 123, 368, 159, 213, 410, 159, 236,
127, 159, 21, 373, 184, 424, 327, 250, 176, 176, 175, 284, 316, 176,
284, 327, 111, 250, 284, 175, 175, 264, 111, 176, 219, 111, 427, 427,
176, 284, 427, 353, 428, 55, 184, 493, 158, 136, 99, 287, 264, 334,
264, 213, 213, 292, 481, 93, 264, 292, 295, 295, 6, 367, 279, 173,
308, 285, 158, 308, 335, 299, 137, 137, 572, 41, 137, 137, 41, 94,
335, 220, 36, 224, 420, 36, 265, 265, 91, 91, 71, 123, 264, 91, 91,
123, 107, 30, 22, 292, 35, 241, 356, 298, 14, 298, 441, 35, 121, 71,
63, 130, 63, 488, 363, 71, 63, 307, 194, 71, 71, 220, 121, 125, 71,
```

```
      220, 71, 71, 71, 71, 235, 265, 353, 128, 155, 128, 420, 400, 130,
      173, 183, 183, 184, 130, 173, 183, 13, 183, 130, 130, 183, 183, 353,
      353, 183, 242, 183, 183, 306, 324, 324, 321, 306, 321, 6, 6, 128,
      306, 242, 242, 306, 183, 183, 6, 183, 321, 486, 183, 164, 30, 78,
      138, 158, 138, 34, 206, 362, 55, 70, 67, 21, 375, 136, 298, 81, 298,
      298, 298, 230, 121, 30, 230, 311, 240, 311, 311, 158, 204, 136, 136,
      184, 136, 264, 311, 311, 312, 312, 72, 311, 175, 264, 91, 175, 264,
      121, 461, 312, 312, 238, 475, 350, 512, 350, 312, 313, 350, 312, 366,
      294, 30, 253, 253, 253, 388, 158, 388, 22, 388, 22, 388, 103, 321,
      321, 253, 7, 437, 103, 114, 242, 114, 114, 242, 114, 114, 242, 242,
      242, 306, 242, 114, 7, 353, 335, 27, 241, 299, 312, 364, 506, 409,
      94, 462, 230, 462, 243, 230, 175, 175, 462, 461, 230, 428, 426, 175,
      175, 165, 175, 175, 372, 183, 572, 102, 85, 102, 538, 206, 376, 85,
      85, 284, 85, 85, 284, 398, 83, 160, 265, 308, 398, 310, 583, 289,
      279, 273, 285, 490, 490, 211, 292, 292, 158, 398, 30, 220, 169, 368,
      368, 368, 169, 159, 368, 93, 368, 368, 93, 169, 368, 368, 443, 368,
      298, 443, 368, 298, 538, 345, 345, 311, 178, 54, 311, 215, 178, 175,
      222, 264, 475, 264, 264, 475, 478, 289, 63, 236, 63, 299, 231, 296,
      397, 299, 158, 36, 164, 164, 21, 492, 21, 164, 21, 164, 403, 26, 26,
      588, 179, 234, 169, 465, 295, 67, 41, 353, 295, 538, 161, 185, 306,
      323, 68, 420, 323, 82, 241, 241, 36, 53, 493, 301, 292, 241, 250, 63,
      63, 103, 442, 353, 185, 353, 321, 353, 185, 353, 353, 185, 409, 353,
      589, 34, 271, 271, 34, 86, 34, 34, 353, 353, 39, 414, 4, 95, 95, 4,
      225, 95, 4, 121, 30, 552, 136, 159, 159, 514, 159, 159, 54, 514, 206,
      136, 206, 159, 74, 235, 235, 312, 54, 312, 42, 156, 422, 629, 54,
      465, 265, 165, 250, 35, 165, 175, 659, 175, 175, 8, 8, 8, 8, 206,
      206, 206, 50, 435, 206, 432, 230, 230, 234, 230, 94, 299, 299, 285,
      184, 41, 93, 299, 299, 285, 41, 285, 158, 285, 206, 299, 41, 36, 396,
      364, 364, 120, 396, 514, 91, 382, 538, 807, 717, 22, 93, 412, 54,
      215, 54, 298, 308, 148, 298, 148, 298, 308, 102, 656, 6, 148, 745,
      128, 298, 64, 407, 273, 41, 172, 64, 234, 250, 398, 181, 445, 95,
      236, 441, 477, 504, 102, 196, 137, 364, 60, 453, 137, 364, 367, 334,
      364, 299, 196, 397, 630, 589, 589, 196, 646, 337, 235, 128, 128, 343,
      289, 235, 324, 427, 324, 58, 215, 215, 461, 425, 461, 387, 440, 285,
      440, 440, 285, 387, 632, 325, 325, 440, 461, 425, 425, 387, 627, 191,
      285, 440, 308, 55, 219, 280, 308, 265, 538, 183, 121, 30, 236, 206,
      30, 455, 236, 30, 30, 705, 83, 228, 280, 468, 132, 8, 132, 132, 128,
      409, 173, 353, 132, 409, 35, 128, 450, 137, 398, 67, 432, 423, 235,
      235, 388, 306, 93, 93, 452, 300, 190, 13, 452, 388, 30, 452, 13, 30,
      13, 30, 306, 362, 234, 721, 635, 809, 784, 67, 498, 498, 67, 353,
      635, 67, 183, 159, 445, 285, 183, 53, 183, 445, 265, 432, 57, 420,
      432, 420, 477, 327, 55, 60, 105, 183, 218, 104, 104, 475, 239, 582,
      151, 239, 104, 732, 41, 26, 784, 86, 300, 215, 36, 64, 86, 86, 675,
      294, 64, 86, 528, 550, 493, 565, 298, 230, 312, 295, 538, 298, 295,
      230, 54, 374, 516, 441, 54, 54, 323, 401, 401, 382, 159, 837, 159,
      54, 401, 592, 159, 401, 417, 610, 264, 150, 323, 452, 185, 323, 323,
      185, 403, 185, 423, 165, 425, 219, 407, 270, 231, 99, 93, 231, 631,
      756, 71, 364, 434, 213, 86, 102, 434, 102, 86, 23, 71, 335, 164, 323,
```

```
     409, 381, 4, 124, 41, 424, 206, 41, 124, 41, 41, 703, 635, 124, 493,
     41, 41, 487, 492, 124, 175, 124, 261, 600, 488, 261, 488, 261, 206,
     677, 261, 308, 723, 908, 704, 691, 723, 488, 488, 441, 136, 476, 312,
     136, 550, 572, 728, 550, 22, 312, 312, 22, 55, 413, 183, 280, 593,
     191, 36, 36, 427, 36, 695, 592, 19, 544, 13, 468, 13, 544, 72, 437,
     321, 266, 461, 266, 441, 230, 409, 93, 521, 521, 345, 235, 22, 142,
     150, 102, 569, 235, 264, 91, 521, 264, 7, 102, 7, 498, 521, 235, 537,
     235, 6, 241, 420, 420, 631, 41, 527, 103, 67, 337, 62, 264, 527, 131,
     67, 174, 263, 264, 36, 36, 263, 581, 253, 465, 160, 286, 91, 160, 55,
     4, 4, 631, 631, 608, 365, 465, 294, 427, 427, 335, 669, 669, 129, 93,
     93, 93, 93, 93, 74, 66, 758, 504, 347, 130, 505, 504, 143, 505, 550, 222,
     13, 352, 529, 291, 538, 50, 68, 269, 130, 295, 130, 511, 295, 295,
     130, 486, 132, 61, 206, 185, 368, 669, 22, 175, 492, 207, 373, 452,
     432, 327, 89, 550, 496, 611, 527, 89, 527, 496, 550, 516, 516, 91,
     136, 538, 264, 264, 124, 264, 264, 264, 264, 264, 535, 264, 150, 285,
     398, 285, 582, 398, 475, 81, 694, 694, 64, 81, 694, 234, 607, 723,
     513, 234, 64, 581, 64, 124, 64, 607, 234, 723, 717, 367, 64, 513,
     607, 488, 183, 488, 450, 183, 550, 286, 183, 363, 286, 414, 67, 449,
     449, 366, 215, 235, 95, 295, 295, 41, 335, 21, 445, 225, 21, 295,
     372, 749, 461, 53, 481, 397, 427, 427, 427, 714, 481, 714, 427, 717,
     165, 245, 486, 415, 245, 415, 486, 274, 415, 441, 456, 300, 548, 300,
     422, 422, 757, 11, 74, 430, 430, 136, 409, 430, 749, 191, 819, 592,
     136, 364, 465, 231, 231, 918, 160, 589, 160, 160, 465, 465, 231, 157,
     538, 538, 259, 538, 326, 22, 22, 22, 179, 22, 22, 550, 179, 287, 287,
     417, 327, 498, 498, 287, 488, 327, 538, 488, 583, 488, 287, 335, 287,
     335, 287, 41, 287, 335, 287, 327, 441, 335, 287, 488, 538, 327, 498,
     8, 8, 374, 8, 64, 427, 8, 374, 417, 760, 409, 373, 160, 423, 206,
     160, 106, 499, 160, 271, 235, 160, 590, 353, 695, 478, 619, 590, 353,
     13, 63, 189, 420, 605, 427, 643, 121, 280, 415, 121, 415, 595, 417,
     121, 398, 55, 330, 463, 463, 123, 353, 330, 582, 309, 582, 582, 405,
     330, 550, 405, 582, 353, 309, 308, 60, 353, 7, 60, 71, 353, 189, 183,
     183, 183, 582, 755, 189, 437, 287, 189, 183, 668, 481, 384, 384, 481,
     481, 481, 477, 582, 582, 499, 650, 481, 121, 461, 231, 36, 235, 36,
     413, 235, 209, 36, 689, 114, 353, 353, 235, 592, 36, 353, 413, 209,
     70, 308, 70, 699, 308, 70, 213, 292, 86, 689, 465, 55, 508, 128, 452,
     29, 41, 681, 573, 352, 21, 21, 648, 648, 69, 509, 409, 21, 264, 21,
     509, 514, 514, 409, 21, 264, 443, 443, 427, 160, 433, 663, 433, 231,
     646, 185, 482, 646, 433, 13, 398, 172, 234, 42, 491, 172, 234, 234,
     832, 775, 172, 196, 335, 822, 461, 298, 461, 364, 1120, 537, 169,
     169, 364, 694, 219, 612, 231, 740, 42, 235, 321, 279, 960, 279, 353,
     492, 159, 572, 321, 159, 287, 353, 287, 287, 206, 206, 321, 287, 159,
     321, 492, 159, 55, 572, 600, 270, 492, 784, 173, 91, 91, 443, 443,
     582, 261, 497, 572, 91, 555, 352, 206, 261, 555, 285, 91, 555, 497,
     83, 91, 619, 353, 488, 112, 4, 592, 295, 295, 488, 235, 231, 769,
     568, 581, 671, 451, 451, 483, 299, 1011, 432, 422, 207, 106, 701,
     508, 555, 508, 555, 125, 870, 555, 589, 508, 125, 749, 482, 125, 125,
     130, 544, 643, 643, 544, 488, 22, 643, 130, 335, 544, 22, 130, 544,
     544, 488, 426, 426, 4, 180, 4, 695, 35, 54, 433, 500, 592, 433, 262,
```

```
94, 401, 401, 106, 216, 216, 106, 521, 102, 462, 518, 271, 475, 365,
193, 648, 206, 424, 206, 193, 206, 206, 424, 299, 590, 590, 364, 621,
67, 538, 488, 567, 51, 51, 513, 194, 81, 488, 486, 289, 567, 563,
749, 563, 338, 338, 502, 563, 822, 338, 563, 338, 502, 201, 230, 201,
533, 445, 175, 201, 175, 13, 85, 960, 103, 85, 175, 30, 445, 445,
175, 573, 196, 877, 287, 356, 678, 235, 489, 312, 572, 264, 717, 138,
295, 6, 295, 523, 55, 165, 165, 295, 138, 663, 6, 295, 6, 353, 138,
6, 138, 169, 129, 784, 12, 129, 194, 605, 784, 445, 234, 627, 563,
689, 627, 647, 570, 627, 570, 647, 206, 234, 215, 234, 816, 627, 816,
234, 627, 215, 234, 627, 264, 427, 427, 30, 424, 161, 161, 916, 740,
180, 616, 481, 514, 383, 265, 481, 164, 650, 121, 582, 689, 420, 669,
589, 420, 788, 549, 165, 734, 280, 224, 146, 681, 788, 184, 398, 784,
4, 398, 417, 417, 398, 636, 784, 417, 81, 398, 417, 81, 185, 827,
420, 241, 420, 41, 185, 185, 718, 241, 101, 185, 185, 241, 241, 241,
241, 241, 185, 324, 420, 420, 1011, 420, 827, 241, 184, 563, 241,
183, 285, 529, 285, 808, 822, 891, 822, 488, 285, 486, 619, 55, 869,
39, 567, 39, 289, 203, 158, 289, 710, 818, 158, 818, 355, 29, 409,
203, 308, 648, 792, 308, 308, 91, 308, 6, 592, 792, 106, 106, 308,
41, 178, 91, 751, 91, 259, 734, 166, 36, 327, 166, 230, 205, 205,
172, 128, 230, 432, 623, 838, 623, 432, 278, 432, 42, 916, 432, 694,
623, 352, 452, 93, 314, 93, 93, 641, 88, 970, 914, 230, 61, 159, 270,
159, 493, 159, 755, 159, 409, 30, 30, 836, 128, 241, 99, 102, 984,
538, 102, 102, 273, 639, 838, 102, 102, 136, 637, 508, 627, 285, 465,
327, 327, 21, 749, 327, 749, 21, 845, 21, 21, 409, 749, 1367, 806,
616, 714, 253, 616, 714, 714, 112, 375, 21, 112, 375, 375, 51, 51,
51, 51, 393, 206, 870, 713, 193, 802, 21, 1061, 42, 382, 42, 543,
876, 42, 876, 382, 696, 543, 635, 490, 353, 353, 417, 64, 1257, 271,
64, 377, 127, 127, 537, 417, 905, 353, 538, 465, 605, 876, 427, 324,
514, 852, 427, 53, 427, 557, 173, 173, 7, 1274, 563, 31, 31, 31, 745,
392, 289, 230, 230, 230, 91, 218, 327, 420, 420, 128, 901, 552, 420,
230, 608, 552, 476, 347, 476, 231, 159, 137, 716, 648, 716, 627, 740,
718, 679, 679, 6, 718, 740, 6, 189, 679, 125, 159, 757, 1191, 409,
175, 250, 409, 67, 324, 681, 605, 550, 398, 550, 931, 478, 174, 21,
316, 91, 316, 654, 409, 425, 425, 699, 61, 699, 321, 698, 321, 698,
61, 425, 699, 321, 409, 699, 299, 335, 321, 335, 61, 698, 699, 654,
698, 299, 425, 231, 14, 121, 515, 121, 14, 165, 81, 409, 189, 81,
373, 465, 463, 1055, 507, 81, 81, 189, 1246, 321, 409, 886, 104, 842,
689, 300, 740, 380, 656, 656, 832, 656, 380, 300, 300, 206, 187, 175,
142, 465, 206, 271, 468, 215, 560, 83, 215, 83, 215, 215, 83, 175,
215, 83, 83, 111, 206, 756, 559, 756, 1367, 206, 559, 1015, 559, 559,
946, 1015, 548, 559, 756, 1043, 756, 698, 159, 414, 308, 458, 997,
663, 663, 347, 39, 755, 838, 323, 755, 323, 159, 159, 717, 159, 21,
41, 128, 516, 159, 717, 71, 870, 755, 159, 740, 717, 374, 516, 740,
51, 148, 335, 148, 335, 791, 120, 364, 335, 335, 51, 120, 251, 538,
251, 971, 1395, 538, 78, 178, 538, 538, 918, 129, 918, 129, 538, 538,
656, 129, 538, 538, 129, 538, 1051, 538, 128, 838, 931, 998, 823,
1095, 334, 870, 334, 367, 550, 1061, 498, 745, 832, 498, 745, 716,
498, 498, 128, 997, 832, 716, 832, 130, 642, 616, 497, 432, 432, 432,
```

432, 642, 159, 432, 46, 230, 788, 160, 230, 478, 46, 693, 103, 920,
230, 589, 643, 160, 616, 432, 165, 165, 583, 592, 838, 784, 583, 710,
6, 583, 583, 6, 35, 230, 838, 592, 710, 6, 589, 230, 838, 30, 592,
583, 6, 583, 6, 6, 583, 30, 30, 6, 375, 375, 99, 36, 1158, 425, 662,
417, 681, 364, 375, 1025, 538, 822, 669, 893, 538, 538, 450, 409,
632, 527, 632, 563, 632, 527, 550, 71, 698, 550, 39, 550, 514, 537,
514, 537, 111, 41, 173, 592, 173, 648, 173, 173, 173, 1011, 514, 173,
173, 514, 166, 648, 355, 161, 166, 648, 497, 327, 327, 550, 650, 21,
425, 605, 555, 103, 425, 605, 842, 836, 1011, 636, 138, 756, 836,
756, 756, 353, 1011, 636, 636, 1158, 741, 741, 842, 756, 741, 1011,
677, 1011, 770, 366, 306, 488, 920, 920, 665, 775, 502, 500, 775,
775, 648, 364, 833, 207, 13, 93, 500, 364, 500, 665, 500, 93, 295,
183, 1293, 313, 272, 313, 279, 303, 93, 516, 93, 1013, 381, 6, 93,
93, 303, 259, 643, 168, 673, 230, 1261, 230, 230, 673, 1060, 1079,
1079, 550, 741, 741, 590, 527, 741, 741, 442, 741, 442, 848, 741,
590, 925, 219, 527, 925, 335, 442, 590, 239, 590, 590, 590, 239, 527,
239, 1033, 230, 734, 241, 741, 230, 549, 548, 1015, 1015, 32, 36,
433, 465, 724, 465, 73, 73, 73, 465, 808, 73, 592, 1430, 250, 154,
154, 250, 538, 353, 353, 353, 353, 353, 175, 194, 206, 538, 632,
1163, 960, 175, 175, 538, 452, 632, 1163, 175, 538, 960, 194, 175,
194, 632, 960, 632, 94, 632, 461, 960, 1163, 1163, 461, 632, 960,
755, 707, 105, 382, 625, 382, 382, 784, 707, 871, 559, 387, 387, 871,
784, 559, 784, 88, 36, 570, 314, 1028, 975, 335, 335, 398, 573, 573,
573, 21, 215, 562, 738, 612, 424, 21, 103, 788, 870, 912, 23, 186,
757, 73, 818, 23, 73, 563, 952, 262, 563, 137, 262, 1022, 952, 137,
1273, 442, 952, 604, 137, 308, 384, 913, 235, 325, 695, 398, 95, 668,
776, 713, 309, 691, 22, 10, 364, 682, 682, 578, 481, 1252, 1072,
1252, 825, 578, 825, 1072, 1149, 592, 273, 387, 273, 427, 155, 1204,
50, 452, 50, 1142, 50, 367, 452, 1142, 611, 367, 50, 50, 367, 50,
1675, 99, 367, 50, 1501, 1099, 830, 681, 689, 917, 1089, 453, 425,
235, 918, 538, 550, 335, 161, 387, 859, 324, 21, 838, 859, 1123, 21,
723, 21, 335, 335, 206, 21, 364, 1426, 21, 838, 838, 335, 364, 21,
21, 859, 920, 838, 838, 397, 81, 639, 397, 397, 588, 933, 933, 784,
222, 830, 36, 36, 222, 1251, 266, 36, 146, 266, 366, 581, 605, 366,
22, 966, 681, 681, 433, 730, 1013, 550, 21, 21, 938, 488, 516, 21,
21, 656, 420, 323, 323, 323, 327, 323, 918, 581, 581, 830, 361, 830,
364, 259, 364, 496, 496, 364, 691, 705, 691, 475, 427, 1145, 600,
179, 427, 527, 749, 869, 689, 335, 347, 220, 298, 689, 1426, 183,
554, 55, 832, 550, 550, 165, 770, 957, 67, 1386, 219, 683, 683, 355,
683, 355, 355, 738, 355, 842, 931, 266, 325, 349, 256, 1113, 256,
423, 960, 554, 554, 325, 554, 508, 22, 142, 22, 508, 916, 767, 55,
1529, 767, 55, 1286, 93, 972, 550, 931, 1286, 1286, 972, 93, 1286,
1392, 890, 93, 1286, 93, 1286, 972, 374, 931, 890, 808, 779, 975,
975, 175, 173, 4, 681, 383, 1367, 173, 383, 1367, 383, 173, 175, 69,
238, 146, 238, 36, 148, 888, 238, 173, 238, 148, 238, 888, 185, 925,
925, 797, 925, 815, 925, 469, 784, 289, 784, 925, 797, 925, 925,
1093, 925, 925, 925, 1163, 797, 797, 815, 925, 1093, 784, 636, 663,
925, 187, 922, 316, 1380, 709, 916, 916, 187, 355, 948, 916, 187,

```
916, 916, 948, 948, 916, 355, 316, 316, 334, 300, 1461, 36, 583,
1179, 699, 235, 858, 583, 699, 858, 699, 1189, 1256, 1189, 699, 797,
699, 699, 699, 699, 427, 488, 427, 488, 175, 815, 656, 656, 150, 322,
465, 322, 870, 465, 1099, 582, 665, 767, 749, 635, 749, 600, 1448,
36, 502, 235, 502, 355, 502, 355, 355, 355, 172, 355, 355, 95, 866,
425, 393, 1165, 42, 42, 42, 393, 939, 909, 909, 836, 552, 424, 1333,
852, 897, 1426, 1333, 1446, 1426, 997, 1011, 852, 1198, 55, 32, 239,
588, 681, 681, 239, 1401, 32, 588, 239, 462, 286, 1260, 984, 1160,
960, 960, 486, 828, 462, 960, 1199, 581, 850, 663, 581, 751, 581,
581, 1571, 252, 252, 1283, 264, 430, 264, 430, 430, 842, 252, 745,
21, 307, 681, 1592, 488, 857, 857, 1161, 857, 857, 857, 138, 374,
374, 1196, 374, 1903, 1782, 1626, 414, 112, 1477, 1040, 356, 775,
414, 414, 112, 356, 775, 435, 338, 1066, 689, 689, 1501, 689, 1249,
205, 689, 765, 220, 308, 917, 308, 308, 220, 327, 387, 838, 917, 917,
917, 220, 662, 308, 220, 387, 387, 220, 220, 308, 308, 308, 387,
1009, 1745, 822, 279, 554, 1129, 543, 383, 870, 1425, 241, 870, 241,
383, 716, 592, 21, 21, 592, 425, 550, 550, 550, 427, 230, 57, 483,
784, 860, 57, 308, 57, 486, 870, 447, 486, 433, 433, 870, 433, 997,
486, 443, 433, 433, 997, 486, 1292, 47, 708, 81, 895, 394, 81, 935,
81, 81, 81, 374, 986, 916, 1103, 1095, 465, 495, 916, 667, 1745, 518,
220, 1338, 220, 734, 1294, 741, 166, 828, 741, 741, 1165, 1371, 1371,
471, 1371, 647, 1142, 1878, 1878, 1371, 1371, 822, 66, 327, 158, 427,
427, 465, 465, 676, 676, 30, 30, 676, 676, 893, 1592, 93, 455, 308,
582, 695, 582, 629, 582, 85, 1179, 85, 85, 1592, 1179, 280, 1027,
681, 398, 1027, 398, 295, 784, 740, 509, 425, 968, 509, 46, 833, 842,
401, 184, 401, 464, 6, 1501, 1501, 550, 538, 883, 538, 883, 883, 883,
1129, 550, 550, 333, 689, 948, 21, 21, 241, 2557, 2094, 273, 308, 58,
863, 893, 1086, 409, 136, 1086, 592, 592, 830, 830, 883, 830, 277,
68, 689, 902, 277, 453, 507, 129, 689, 630, 664, 550, 128, 1626,
1626, 128, 902, 312, 589, 755, 755, 589, 755, 407, 1782, 589, 784,
1516, 1118, 407, 407, 1447, 589, 235, 755, 1191, 235, 235, 407, 128,
589, 1118, 21, 383, 1331, 691, 481, 383, 1129, 1129, 1261, 1104,
1378, 1129, 784, 1129, 1261, 1129, 947, 1129, 784, 784, 1129, 1129,
35, 1104, 35, 866, 1129, 1129, 64, 481, 730, 1260, 481, 970, 481,
481, 481, 481, 863, 481, 681, 699, 863, 486, 681, 481, 481, 55, 55,
235, 1364, 944, 632, 822, 401, 822, 952, 822, 822, 99, 550, 2240,
550, 70, 891, 860, 860, 550, 550, 916, 1176, 1530, 425, 1530, 916,
628, 1583, 916, 628, 916, 916, 628, 628, 425, 916, 1062, 1265, 916,
916, 916, 280, 461, 916, 916, 1583, 628, 1062, 916, 916, 677, 1297,
924, 1260, 83, 1260, 482, 433, 234, 462, 323, 1656, 997, 323, 323,
931, 838, 931, 1933, 1391, 367, 323, 931, 1391, 1391, 103, 1116,
1116, 1116, 769, 1195, 1218, 312, 791, 312, 741, 791, 997, 312, 334,
334, 312, 287, 287, 633, 1397, 1426, 605, 1431, 327, 592, 705, 1194,
592, 1097, 1118, 1503, 1267, 1267, 1267, 618, 1229, 734, 1089, 785,
1089, 1129, 1148, 1148, 1089, 915, 1148, 1129, 1148, 1011, 1011,
1229, 871, 1560, 1560, 1560, 563, 1537, 1009, 1560, 632, 985, 592,
1308, 592, 882, 145, 145, 397, 837, 383, 592, 592, 832, 36, 2714,
2107, 1588, 1347, 36, 36, 1443, 1453, 334, 2230, 1588, 1169, 650,
```

```
   1169, 2107, 425, 425, 891, 891, 425, 2532, 679, 274, 274, 274, 325,
   274, 1297, 194, 1297, 627, 314, 917, 314, 314, 1501, 414, 1490, 1036,
   592, 1036, 1025, 901, 1218, 1025, 901, 280, 592, 592, 901, 1461, 159,
   159, 159, 2076, 1066, 1176, 1176, 516, 327, 516, 1179, 1176, 899,
   1176, 1176, 323, 1187, 1229, 663, 1229, 504, 1229, 916, 1229, 916,
   1661, 41, 36, 278, 1027, 648, 648, 648, 1626, 648, 646, 1179, 1580,
   1061, 1514, 1008, 1741, 2076, 1514, 1008, 952, 1089, 427, 952, 427,
   1083, 425, 427, 1089, 1083, 425, 427, 425, 230, 920, 1678, 920, 1678,
   189, 189, 953, 189, 133, 189, 1075, 189, 189, 133, 1264, 725, 189,
   1629, 189, 808, 230, 230, 2179, 770, 230, 770, 230, 21, 21, 784,
   1118, 230, 230, 230, 770, 1118, 986, 808, 916, 30, 327, 918, 679,
   414, 916, 1165, 1355, 916, 755, 733, 433, 1490, 433, 433, 433, 605,
   433, 433, 433, 1446, 679, 206, 433, 21, 2452, 206, 206, 433, 1894,
   206, 822, 206, 2073, 206, 206, 21, 822, 21, 206, 206, 21, 383, 1513,
   375, 1347, 432, 1589, 172, 954, 242, 1256, 1256, 1248, 1256, 1256,
   1248, 1248, 1256, 842, 13, 592, 13, 842, 1291, 592, 21, 175, 13, 592,
   13, 13, 1426, 13, 1541, 445, 808, 808, 863, 647, 219, 1592, 1029,
   1225, 917, 1963, 1129, 555, 1313, 550, 660, 550, 220, 660, 552, 663,
   220, 533, 220, 383, 550, 1278, 1495, 636, 842, 1036, 425, 842, 425,
   1537, 1278, 842, 554, 1508, 636, 554, 301, 842, 792, 1392, 1021, 284,
   1172, 997, 1021, 103, 1316, 308, 1210, 848, 848, 1089, 1089, 848,
   848, 67, 1029, 827, 1029, 2078, 827, 1312, 1029, 827, 590, 872, 1312,
   427, 67, 67, 67, 67, 872, 827, 872, 2126, 1436, 26, 2126, 67, 1072,
   2126, 1610, 872, 1620, 883, 883, 1397, 1189, 555, 555, 563, 1189,
   555, 640, 555, 640, 1089, 1089, 610, 610, 1585, 610, 1355, 610, 1015,
   616, 925, 1015, 482, 230, 707, 231, 888, 1355, 589, 1379, 151, 931,
   1486, 1486, 393, 235, 960, 590, 235, 960, 422, 142, 285, 285, 327,
   327, 442, 2009, 822, 445, 822, 567, 888, 2611, 1537, 323, 55, 1537,
   323, 888, 2611, 323, 1537, 323, 58, 445, 593, 2045, 593, 58, 47, 770,
   842, 47, 47, 842, 842, 648, 2557, 173, 689, 2291, 1446, 2085, 2557,
   2557, 2291, 1780, 1535, 2291, 2391, 808, 691, 1295, 1165, 983, 948,
   2000, 948, 983, 983, 2225, 2000, 983, 983, 705, 948, 2000, 1795,
   1592, 478, 592, 1795, 1795, 663, 478, 1790, 478, 592, 1592, 173, 901,
   312, 4, 1606, 173, 838, 754, 754, 128, 550, 1166, 551, 1480, 550,
   550, 1875, 1957, 1166, 902, 1875, 550, 550, 551, 2632, 551, 1875,
   1875, 551, 2891, 2159, 2632, 3231, 551, 815, 150, 1654, 1059, 1059,
   734, 770, 555, 1592, 555, 2059, 770, 770, 1803, 627, 627, 627, 2059,
   931, 1272, 427, 1606, 1272, 1606, 1187, 1204, 397, 822, 21, 1645,
   263, 263, 822, 263, 1645, 280, 263, 605, 1645, 2014, 21, 21, 1029,
   263, 1916, 2291, 397, 397, 496, 270, 270, 1319, 264, 1638, 264, 986,
   1278, 1397, 1278, 1191, 409, 1191, 740, 1191, 754, 754, 387, 63, 948,
   666, 666, 1198, 548, 63, 1248, 285, 1248, 169, 1248, 1248, 285, 918,
   224, 285, 1426, 1671, 514, 514, 717, 514, 51, 1521, 1745, 51, 605,
   1191, 51, 128, 1191, 51, 51, 1521, 267, 513, 952, 966, 1671, 897, 51,
   71, 592, 986, 986, 1121, 592, 280, 2000, 2000, 1165, 1165, 1165,
   1818, 222, 1818, 1165, 1252, 506, 327, 443, 432, 1291, 1291, 2755,
   1413, 520, 1318, 227, 1047, 828, 520, 347, 1364, 136, 136, 452, 457,
   457, 132, 457, 488, 1087, 1013, 2225, 32, 1571, 2009, 483, 67, 483,
```

       740, 740, 1013, 2854, 866, 32, 2861, 866, 887, 32, 2444, 740, 32, 32,
       866, 2225, 866, 32, 1571, 2627, 32, 850, 1675, 569, 1158, 32, 1158,
       1797, 2641, 1565, 1158, 569, 1797, 1158, 1797, 55, 1703, 42, 55,
       2562, 675, 1703, 42, 55, 749, 488, 488, 347, 1206, 1286, 1286, 488,
       488, 1206, 1286, 1206, 1286, 550, 550, 1790, 860, 550, 2452, 550,
       550, 2765, 1089, 1633, 797, 2244, 1313, 194, 2129, 194, 194, 194,
       818, 32, 194, 450, 1313, 2387, 194, 1227, 2387, 308, 2232, 526, 476,
       278, 830, 830, 194, 830, 194, 278, 194, 714, 476, 830, 714, 830, 278,
       830, 2532, 1218, 1759, 1446, 960, 1747, 187, 1446, 1759, 960, 105,
       1446, 1446, 1271, 1446, 960, 960, 1218, 1446, 1446, 105, 1446, 960,
       488, 1446, 427, 534, 842, 1969, 2460, 1969, 842, 842, 1969, 427, 941,
       2160, 427, 230, 938, 2075, 1675, 1675, 895, 1675, 34, 129, 1811, 239,
       749, 1957, 2271, 749, 1908, 129, 239, 239, 129, 129, 2271, 2426,
       1355, 1756, 194, 1583, 194, 194, 1583, 194, 1355, 194, 1628, 2221,
       1269, 2425, 1756, 1355, 1355, 1583, 1033, 427, 582, 30, 582, 582,
       935, 1444, 1962, 915, 733, 915, 938, 1962, 767, 353, 1630, 1962,
       1962, 563, 733, 563, 733, 353, 822, 1630, 740, 2076, 2076, 2076, 589,
       589, 2636, 866, 589, 947, 1528, 125, 273, 1058, 1058, 1161, 1635,
       1355, 1161, 1161, 1355, 1355, 650, 1206, 1206, 784, 784, 784, 784,
       784, 412, 461, 412, 2240, 412, 679, 891, 461, 679, 679, 189, 189,
       1933, 1651, 2515, 189, 1386, 538, 1386, 1386, 1187, 1386, 2423, 2601,
       2285, 175, 175, 2331, 194, 3079, 384, 538, 2365, 2294, 538, 2166,
       1841, 3326, 1256, 3923, 976, 85, 550, 550, 1295, 863, 863, 550, 1249,
       550, 1759, 146, 1069, 920, 2633, 885, 885, 1514, 1489, 166, 1514,
       2041, 885, 2456, 885, 2041, 1081, 1948, 362, 550, 94, 324, 2308, 94,
       2386, 94, 550, 874, 1329, 1759, 2280, 1487, 493, 493, 2099, 2599,
       1431, 1086, 1514, 1086, 2099, 1858, 368, 1330, 2599, 1858, 2846,
       2846, 2907, 2846, 713, 713, 1854, 1123, 713, 713, 3010, 1123, 3010,
       538, 713, 1123, 447, 822, 555, 2011, 493, 508, 2292, 555, 1736, 2135,
       2704, 555, 2814, 555, 2000, 555, 555, 822, 914, 327, 679, 327, 648,
       537, 2263, 931, 1496, 537, 1296, 1745, 1592, 1658, 1795, 650, 1592,
       1745, 1745, 1658, 1592, 1745, 1592, 1745, 1658, 1338, 2124, 1592,
       1745, 1745, 1745, 837, 1726, 2897, 1118, 1118, 230, 1118, 1118, 1118,
       1388, 1748, 514, 128, 1165, 931, 514, 2974, 2041, 2387, 2041, 979,
       185, 36, 1269, 550, 173, 812, 36, 1165, 2676, 2562, 1473, 2885, 1982,
       1578, 1578, 383, 383, 2360, 383, 1578, 2360, 1584, 1982, 1578, 1578,
       1578, 2019, 1036, 355, 724, 2023, 205, 303, 355, 1036, 1966, 355,
       1036, 401, 401, 401, 830, 401, 849, 578, 401, 849, 849, 578, 1776,
       1123, 552, 2632, 808, 1446, 1120, 373, 1529, 1483, 1057, 893, 1284,
       1430, 1529, 1529, 2632, 1352, 2063, 1606, 1352, 1606, 2291, 3079,
       2291, 1529, 506, 838, 1606, 1606, 1352, 1529, 1529, 1483, 1529, 1606,
       1529, 259, 902, 259, 902, 612, 612, 284, 398, 2991, 1534, 1118, 1118,
       1118, 1118, 1118, 734, 284, 2224, 398, 734, 284, 734, 398, 3031, 398,
       734, 1707, 2643, 1344, 1477, 475, 1818, 194, 1894, 691, 1528, 1184,
       1207, 1501, 6, 2069, 871, 2069, 3548, 1443, 2069, 2685, 3265, 1350,
       3265, 2069, 2069, 128, 1313, 128, 663, 414, 1313, 414, 2000, 128,
       2000, 663, 1313, 699, 1797, 550, 327, 550, 1526, 699, 327, 1797,
       1526, 550, 550, 327, 550, 1426, 1426, 1426, 2285, 1123, 890, 728,

     1707, 728, 728, 327, 253, 1187, 1281, 1364, 1571, 2170, 755, 3232,
     925, 1496, 2170, 2170, 1125, 443, 902, 902, 925, 755, 2078, 2457,
     902, 2059, 2170, 1643, 1129, 902, 902, 1643, 1129, 606, 36, 103, 338,
     338, 1089, 338, 338, 338, 1089, 338, 36, 340, 1206, 1176, 2041, 833,
     1854, 1916, 1916, 1501, 2132, 1736, 3065, 367, 1934, 833, 833, 833,
     2041, 3017, 2147, 818, 1397, 828, 2147, 398, 828, 818, 1158, 818,
     689, 327, 36, 1745, 2132, 582, 1475, 189, 582, 2132, 1191, 582, 2132,
     1176, 1176, 516, 2610, 2230, 2230, 64, 1501, 537, 1501, 173, 2230,
     2988, 1501, 2694, 2694, 537, 537, 173, 173, 1501, 537, 64, 173, 173,
     64, 2230, 537, 2230, 537, 2230, 2230, 2069, 3142, 1645, 689, 1165,
     1165, 1963, 514, 488, 1963, 1145, 235, 1145, 1078, 1145, 231, 2405,
     552, 21, 57, 57, 57, 1297, 1455, 1988, 2310, 1885, 2854, 2014, 734,
     1705, 734, 2854, 734, 677, 1988, 1660, 734, 677, 734, 677, 677, 734,
     2854, 1355, 677, 1397, 2947, 2386, 1698, 128, 1698, 3028, 2386, 2437,
     2947, 2386, 2643, 2386, 2804, 1188, 335, 746, 1187, 1187, 861, 2519,
     1917, 2842, 1917, 675, 1308, 234, 1917, 314, 314, 2339, 2339, 2592,
     2576, 902, 916, 2339, 916, 2339, 916, 2339, 916, 1089, 1089, 2644,
     1221, 1221, 2446, 308, 308, 2225, 2225, 3192, 2225, 555, 1592, 1592,
     555, 893, 555, 550, 770, 3622, 2291, 2291, 3419, 465, 250, 2842,
     2291, 2291, 2291, 935, 160, 1271, 308, 325, 935, 1799, 1799, 1891,
     2227, 1799, 1598, 112, 1415, 1840, 2014, 1822, 2014, 677, 1822, 1415,
     1415, 1822, 2014, 2386, 2159, 1822, 1415, 1822, 179, 1976, 1033, 179,
     1840, 2014, 1415, 1970, 1970, 1501, 563, 563, 563, 462, 563, 1970,
     1158, 563, 563, 1541, 1238, 383, 235, 1158, 383, 1278, 383, 1898,
     2938, 21, 2938, 1313, 2201, 2059, 423, 2059, 1313, 872, 1313, 2044,
     89, 173, 3327, 1660, 2044, 1623, 173, 1114, 1114, 1592, 1868, 1651,
     1811, 383, 3469, 1811, 1651, 869, 383, 383, 1651, 1651, 3223, 2166,
     3469, 767, 383, 1811, 767, 2323, 3355, 1457, 3341, 2640, 2976, 2323,
     3341, 2323, 2640, 103, 103, 1161, 1080, 2429, 370, 2018, 2854, 2429,
     2166, 2429, 2094, 2207, 871, 1963, 1963, 2023, 2023, 2336, 663, 2893,
     1580, 691, 663, 705, 2046, 2599, 409, 2295, 1118, 2494, 1118, 1950,
     549, 2494, 2453, 2046, 2494, 2453, 2046, 2453, 2046, 409, 1118, 4952,
     2291, 2225, 1894, 1423, 2498, 567, 4129, 1475, 1501, 795, 463, 2084,
     828, 828, 232, 828, 232, 232, 1818, 1818, 666, 463, 232, 220, 220,
     2162, 2162, 833, 4336, 913, 35, 913, 21, 2927, 886, 3037, 383, 886,
     876, 1747, 383, 916, 916, 916, 2927, 916, 1747, 837, 1894, 717, 423,
     481, 1894, 1059, 2262, 3206, 4700, 1059, 3304, 2262, 871, 1831, 871,
     3304, 1059, 1158, 1934, 1158, 756, 1511, 41, 978, 1934, 2603, 720,
     41, 756, 41, 325, 2611, 1158, 173, 1123, 1934, 1934, 1511, 2045,
     2045, 2045, 1423, 3206, 3691, 2512, 3206, 2512, 2000, 1811, 2504,
     2504, 2611, 2437, 2437, 2437, 1455, 893, 150, 2665, 1966, 605, 398,
     2331, 1177, 516, 1962, 4241, 94, 1252, 760, 1292, 1962, 1373, 2000,
     1990, 3684, 42, 1868, 3779, 1811, 1811, 2041, 3010, 5436, 1780, 2041,
     1868, 1811, 1780, 1811, 1868, 1811, 2041, 1868, 1811, 5627, 4274,
     1811, 1868, 4602, 1811, 1811, 1474, 2665, 235, 1474, 2665

6. Security Considerations

   Data delivery can be subject to denial-of-service attacks by
   attackers that send corrupted packets that are accepted as legitimate
   by receivers. This is particularly a concern for multicast delivery
   because a corrupted packet may be injected into the session close to
   the root of the multicast tree, in which case, the corrupted packet
   will arrive at many receivers. This is particularly a concern when
   the code described in this document is used because the use of even
   one corrupted packet containing encoding data may result in the
   decoding of an object that is completely corrupted and unusable. It
   is thus RECOMMENDED that source authentication and integrity checking
   are applied to decoded objects before delivering objects to an
   application. For example, a SHA-1 hash [SHA1] of an object may be
   appended before transmission, and the SHA-1 hash is computed and
   checked after the object is decoded but before it is delivered to an
   application. Source authentication SHOULD be provided, for example,
   by including a digital signature verifiable by the receiver computed
   on top of the hash value. It is also RECOMMENDED that a packet
   authentication protocol, such as TESLA [RFC4082], be used to detect
   and discard corrupted packets upon arrival. This method may also be
   used to provide source authentication. Furthermore, it is
   RECOMMENDED that Reverse Path Forwarding checks be enabled in all
   network routers and switches along the path from the sender to
   receivers to limit the possibility of a bad agent successfully
   injecting a corrupted packet into the multicast tree data path.

   Another security concern is that some FEC information may be obtained
   by receivers out-of-band in a session description, and if the session
   description is forged or corrupted, then the receivers will not use
   the correct protocol for decoding content from received packets. To
   avoid these problems, it is RECOMMENDED that measures be taken to
   prevent receivers from accepting incorrect session descriptions,
   e.g., by using source authentication to ensure that receivers only
   accept legitimate session descriptions from authorized senders.

7. IANA Considerations

   Values of FEC Encoding IDs and FEC Instance IDs are subject to IANA
   registration. For general guidelines on IANA considerations as they
   apply to this document, see [RFC5052]. This document assigns the
   Fully-Specified FEC Encoding ID 1 under the ietf:rmt:fec:encoding
   name-space to "Raptor Code".

8. Acknowledgements

   Numerous editorial improvements and clarifications were made to this
   specification during the review process within 3GPP. Thanks are due
   to the members of 3GPP Technical Specification Group SA, Working
   Group 4, for these.

9. References

9.1. Normative References

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC4082]   Perrig, A., Song, D., Canetti, R., Tygar, J., and B.
               Briscoe, "Timed Efficient Stream Loss-Tolerant
               Authentication (TESLA): Multicast Source Authentication
               Transform Introduction", RFC 4082, June 2005.

   [RFC5052]   Watson, M., Luby, M., and L. Vicisano, "Forward Error
               Correction (FEC) Building Block", RFC 5052, August 2007.

9.2. Informative References

   [CCNC]   Luby, M., Watson, M., Gasiba, T., Stockhammer, T., and W.
               Xu, "Raptor Codes for Reliable Download Delivery in
               Wireless Broadcast Systems", CCNC 2006, Las Vegas, NV ,
               Jan 2006.

   [MBMS]   3GPP, "Multimedia Broadcast/Multicast Service (MBMS);
               Protocols and codecs", 3GPP TS 26.346 6.1.0, June 2005.

   [RFC3453]   Luby, M., Vicisano, L., Gemmell, J., Rizzo, L., Handley,
               M., and J. Crowcroft, "The Use of Forward Error Correction
               (FEC) in Reliable Multicast", RFC 3453, December 2002.

   [Raptor]   Shokrollahi, A., "Raptor Codes", IEEE Transactions on
               Information Theory no. 6, June 2006.

   [SHA1]   "Secure Hash Standard", Federal Information Processing
               Standards Publication (FIPS PUB) 180-1, April 2005.

Authors' Addresses

   Michael Luby
   Digital Fountain
   39141 Civic Center Drive
   Suite 300
   Fremont, CA 94538
   U.S.A.

   EMail: luby@digitalfountain.com

   Amin Shokrollahi
   EPFL
   Laboratory of Algorithmic Mathematics
   IC-IIF-ALGO
   PSE-A
   Lausanne 1015
   Switzerland

   EMail: amin.shokrollahi@epfl.ch

   Mark Watson
   Digital Fountain
   39141 Civic Center Drive
   Suite 300
   Fremont, CA 94538
   U.S.A.

   EMail: mark@digitalfountain.com

   Thomas Stockhammer
   Nomor Research
   Brecherspitzstrasse 8
   Munich 81541
   Germany

   EMail: stockhammer@nomor.de