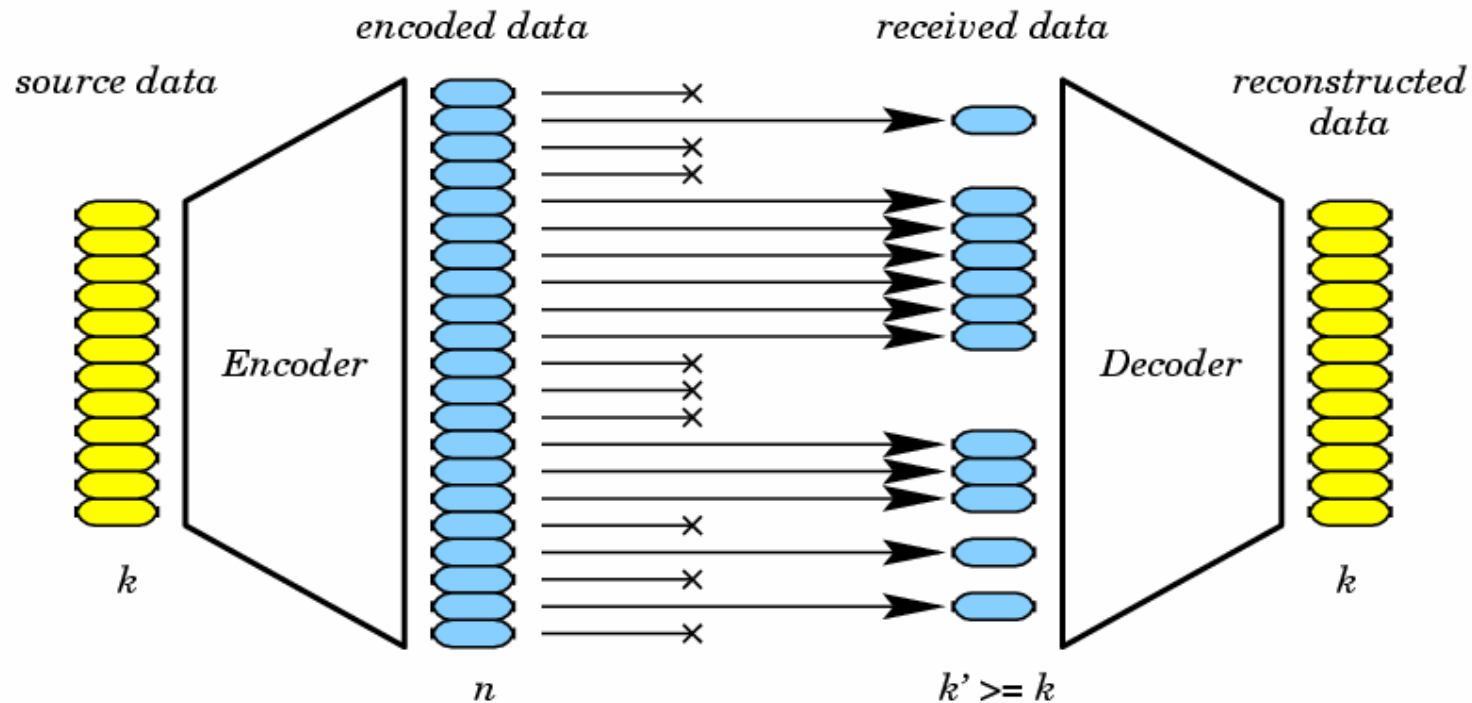


# Reference

L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *ACM SIGCOMM Computer Communication Review*, April 1997

# Encoding/decoding process



Large fixed-length packets can be split into multiple data items for each packet. The encoding/decoding process is applied by taking one data item per packet.

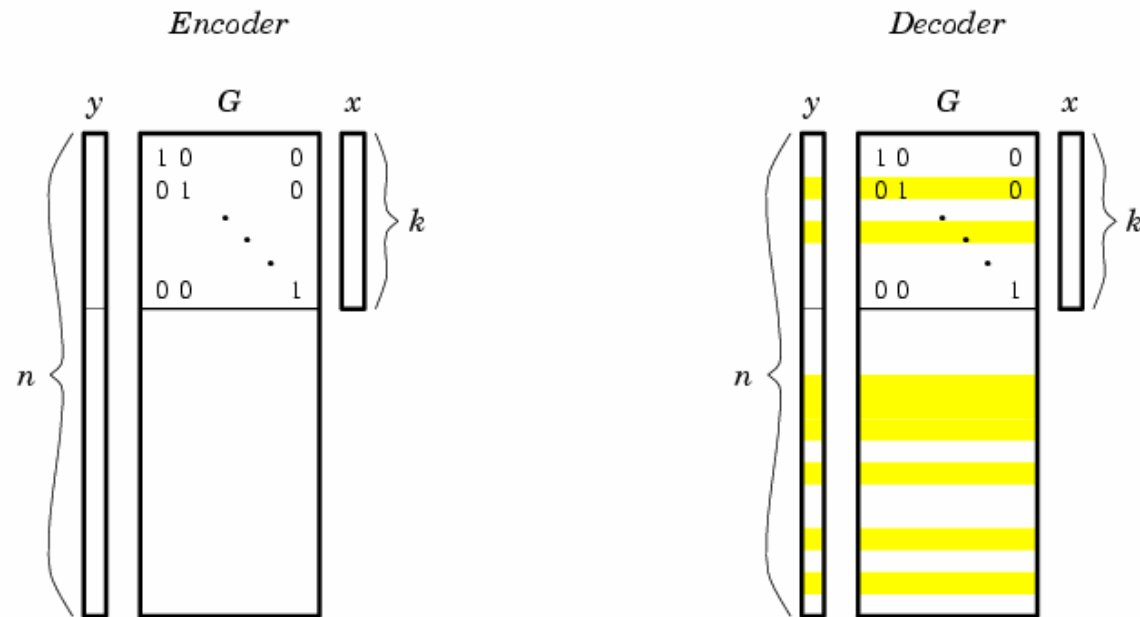
# Linear codes

- Can be analyzed using the properties of linear algebra
- Let  $\underline{x} = x_0 \dots x_{k-1}$  be the source data items,  $G$  an  $n \times k$  matrix, then an  $(n, k)$  linear code can be represented by

$$\underline{y} = G \underline{x}$$

for a properly defined  $G$ ; more specifically, any  $k \times k$  matrix extracted from  $G$  is invertible.

# Encoding/decoding in matrix form



- ❑ For a "systematic" code, the top  $k$  rows of  $G$  constitute the identity matrix.
- ❑ Suppose  $y'$  containing  $k$  components of  $y$  are available at the decoder.  $Y'$  and  $G'$  correspond to the yellow areas of the vector and matrix on the right side.

# Encoding/decoding in matrix form (cont.)

- $G$  is called the generator matrix of the code.
- For a systematic code,  $G$  contains the identity matrix; the remaining rows of the matrix must all contain nonzero elements
- Computations in finite fields
  - A finite field has a finite number of elements. It is closed under additions and multiplications. Thus the sums and products are field elements—exact computations possible without requiring more bits to represent the results.

# RSE coder [Rizzo '97]

$$[p_1, p_2, \dots, p_h]' = G_{h \times k} \cdot [d_1, d_2, \dots, d_k]'$$

$$G_{h \times k} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & \alpha & \dots & \alpha^{k-1} \\ 1 & \alpha^2 & \dots & \alpha^{2(k-1)} \\ \dots & \dots & \dots & \dots \\ 1 & \alpha^{h-1} & \dots & \alpha^{(h-1)(k-1)} \end{bmatrix}$$

- $d_1, d_2, \dots, d_k$ :  $k$  original packets (data items)
- $p_1, p_2, \dots, p_h$ :  $h$  parity packets (data items)
- Consider them to be elements of Galois field  $GF(2^r)$ ,  $r$  ranges from 2 to 16, with 8 being most efficient in Rizzo's implementation
- $\alpha$  is a primitive element in the field; powers of  $\alpha$  generate all elements of the field
- encoding time will increase if  $k$  increases