Decoding a perturbed sequence generated by an LFSR

Sara D. Cardell[1], Joan-Josep Climent[2], Alicia Roca[3]

[1] Universidade Estadual de Campinas, Campinas, Brazil
[2] Universitat d'Alacant, Alacant, Spain
[3] Universitat Politècnica de València, Spain
sdcardell@ime.unicamp.br, jcliment@ua.es, aroca@mat.upv.es

5ICMCTA 2017, 28-31 August 2017, Estonia

Contents

- Introduction. Statement of the problem
- Preliminaries: LFSR. Correlation attacks. Coding theory
- A decoding problem
- Decoding algorithms

Introduction

$\boldsymbol{y} = (y_0, y_1, y_2, \ldots)$ sequence of bits produced by a LFSR

$\boldsymbol{z} = (z_0, z_1, z_2, \ldots)$ perturbation of $\boldsymbol{y} = (y_0, y_1, y_2, \ldots)$

Assume that there exists a **correlation** between $\boldsymbol{z}$ and $\boldsymbol{y}$:

$$Pr(z_i = y_i) = p > 0$$

Problem

Given $\boldsymbol{z}$, knowing the register, find $\boldsymbol{y}$.

Introduction

$\boldsymbol{y} = (y_0, y_1, y_2, \ldots)$ sequence of bits produced by a LFSR

$\boldsymbol{z} = (z_0, z_1, z_2, \ldots)$ perturbation of $\boldsymbol{y} = (y_0, y_1, y_2, \ldots)$

Assume that there exists a **correlation** between $\boldsymbol{z}$ and $\boldsymbol{y}$:

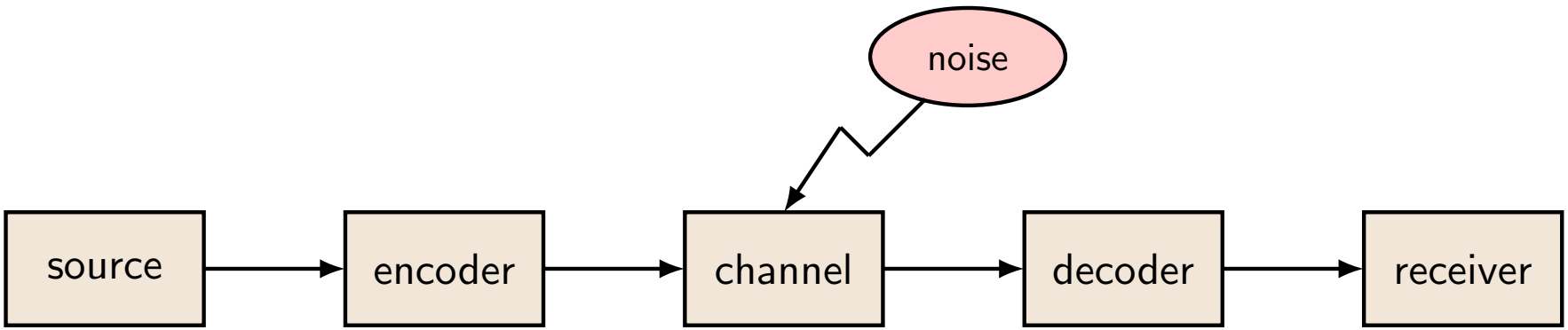
$$Pr(z_i = y_i) = p > 0$$

Problem

Given $\boldsymbol{z}$, knowing the register, find $\boldsymbol{y}$.

Examples of perturbed sequences

A simplified version of a communication channel

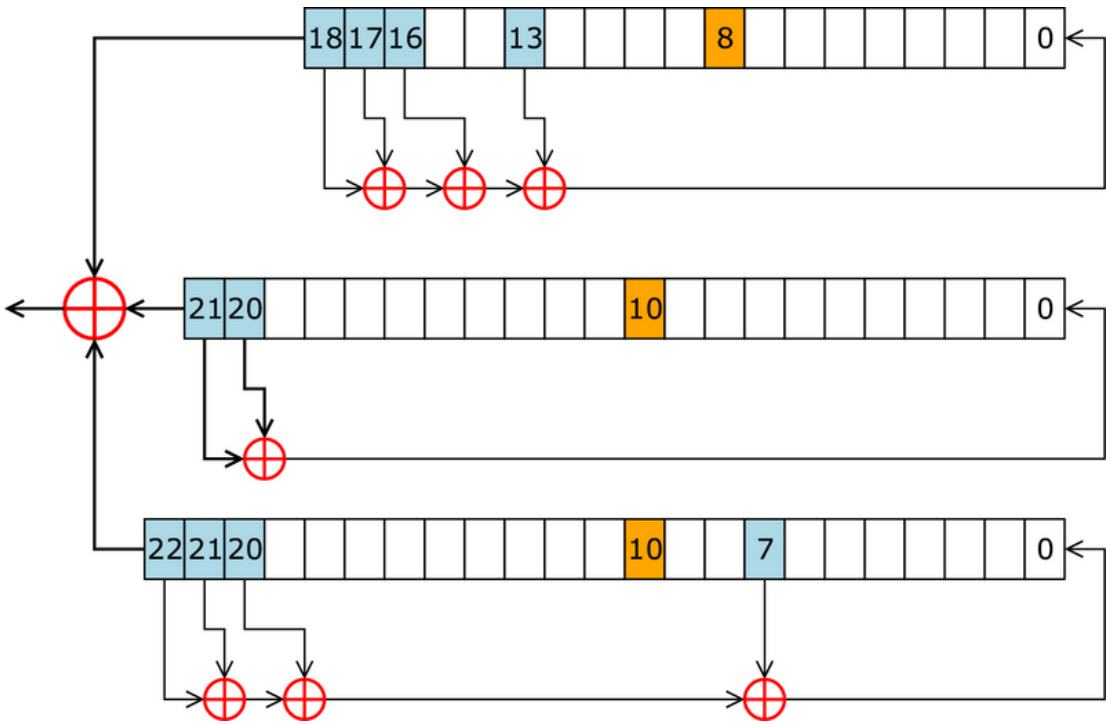


Decoder:

- Inverts the encoding map to recover original source data.
- Attempts to determine if errors ocurred ("error detecting codes"),
- In case of error, tries to correct it and recover the codeword sent ("error correcting codes"),

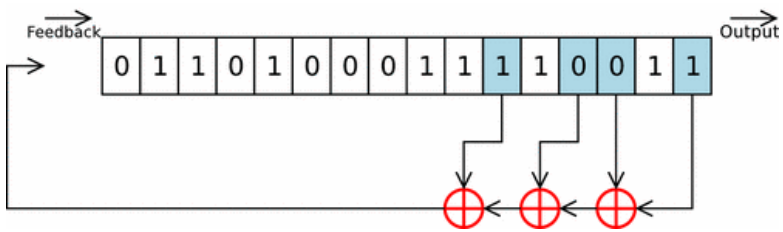
Examples of perturbed sequences

NLFSR: Algorithm A5/1 (mobile communications GSM)



Linear Feedback Shift Registers (LFSR)

- Structures formed by cells of memory (store information: bits).
- Clock control: shifting of the information, producing an output bit.
- The empty cell filled in with the result of a linear feedback function.



$(k, f(x))$
k **length** of the register
feedback: $f(x)$ **connection polynomial**
$f(x) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1} + x^k$
$(y_{L-1}, y_{L-2}, \ldots, y_0)$ **initial state**

In the picture: $f(x) = 1 + x^2 + x^3 + x^5 + x^{16}$

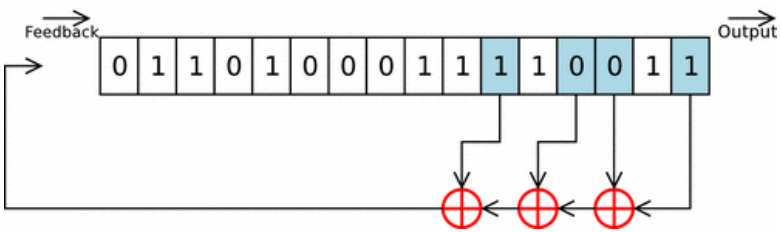
$$y_{16} = y_0 + y_2 + y_3 + y_5$$

In general:

$$y_t = c_0 y_{t-k} + c_1 y_{t-k+1} + c_2 y_{t-k+2} + \cdots + c_{k-2} y_{t-2} + c_{k-1} y_{t-1}, \quad t \geq k$$

Linear Feedback Shift Registers (LFSR)

- Structures formed by cells of memory (store information: bits).
- Clock control: shifting of the information, producing an output bit.
- The empty cell filled in with the result of a linear feedback function.



$(k, f(x))$
k **length** of the register
feedback: $f(x)$ **connection polynomial**
$f(x) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1} + x^k$
$(y_{L-1}, y_{L-2}, \ldots, y_0)$ **initial state**

In the picture: $f(x) = 1 + x^2 + x^3 + x^5 + x^{16}$

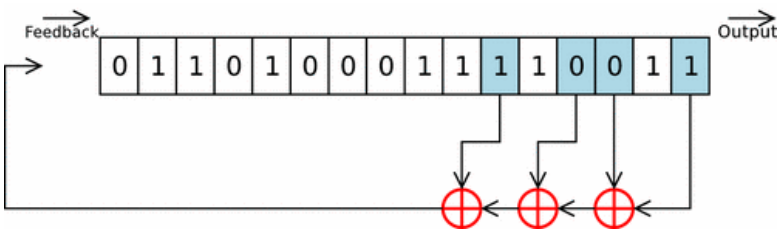
$$y_{16} = y_0 + y_2 + y_3 + y_5$$

In general:

$$y_t = c_0 y_{t-k} + c_1 y_{t-k+1} + c_2 y_{t-k+2} + \cdots + c_{k-2} y_{t-2} + c_{k-1} y_{t-1}, \quad t \geq k$$

Linear Feedback Shift Registers (LFSR)

- Structures formed by cells of memory (store information: bits).
- Clock control: shifting of the information, producing an output bit.
- The empty cell filled in with the result of a linear feedback function.



$(k, f(x))$
k **length** of the register
feedback: $f(x)$ **connection polynomial**
$f(x) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1} + x^k$
$(y_{L-1}, y_{L-2}, \ldots, y_0)$ **initial state**

In the picture: $f(x) = 1 + x^2 + x^3 + x^5 + x^{16}$

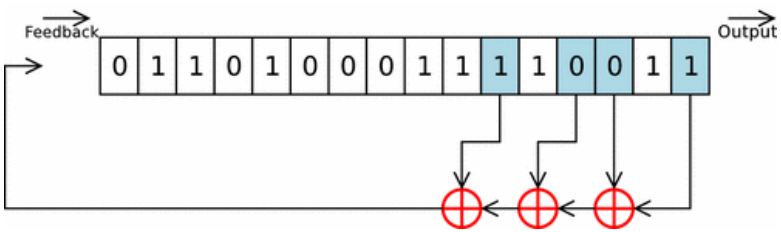
$$y_{16} = y_0 + y_2 + y_3 + y_5$$

In general:

$$y_t = c_0 y_{t-k} + c_1 y_{t-k+1} + c_2 y_{t-k+2} + \cdots + c_{k-2} y_{t-2} + c_{k-1} y_{t-1}, \quad t \geq k$$

Linear Feedback Shift Registers (LFSR)

- Structures formed by cells of memory (store information: bits).
- Clock control: shifting of the information, producing an output bit.
- The empty cell filled in with the result of a linear feedback function.



$(k, f(x))$
k **length** of the register
feedback: $f(x)$ **connection polynomial**
$f(x) = c_0 + c_1 x + \cdots + c_{k-1} x^{k-1} + x^k$
$(y_{L-1}, y_{L-2}, \ldots, y_0)$ **initial state**

In the picture: $f(x) = 1 + x^2 + x^3 + x^5 + x^{16}$

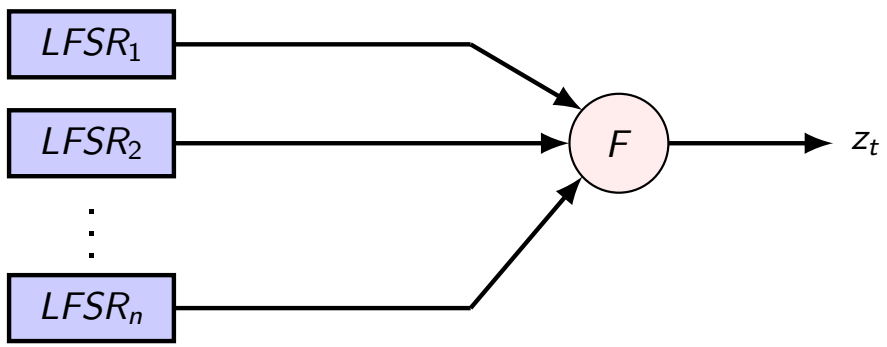
$$y_{16} = y_0 + y_2 + y_3 + y_5$$

In general:

$$y_t = c_0 y_{t-k} + c_1 y_{t-k+1} + c_2 y_{t-k+2} + \cdots + c_{k-2} y_{t-2} + c_{k-1} y_{t-1}, \quad t \geq k$$

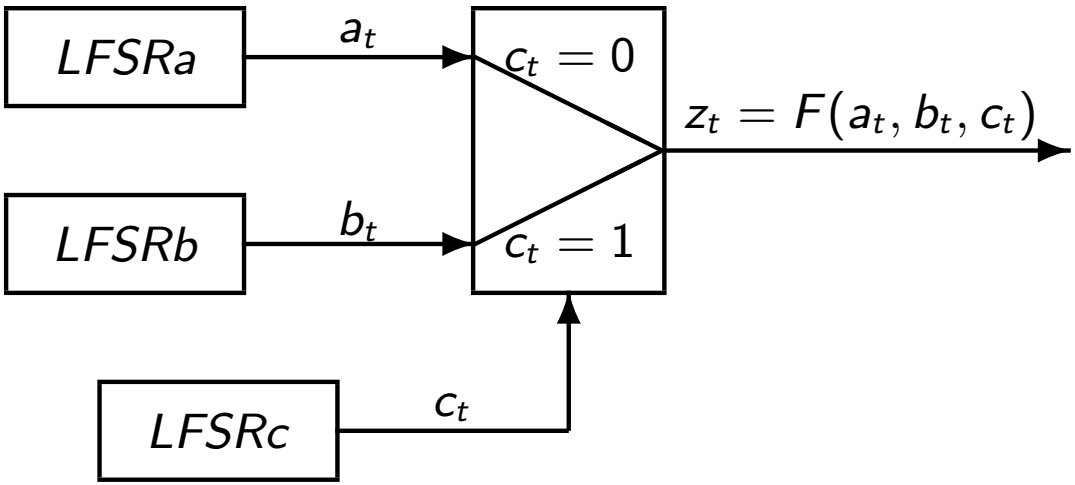
Properties/Remarks

- Knowing $2k$ **bits** of the sequence $\longrightarrow$ easy to compute the feedback polynomial.
- Knowing a **sequence** produced by LFSR $\longrightarrow$ Berlekamp-Massey algorithm finds a LFSR of minimal length generating it.
- To **destroy linearity** $\longrightarrow$ combine several binary LFSR via a nonlinear Boolean function.



- Frequently, the output of the nonlinear implementation is **correlated** with the output of the component registers.

Example: Geffe generator



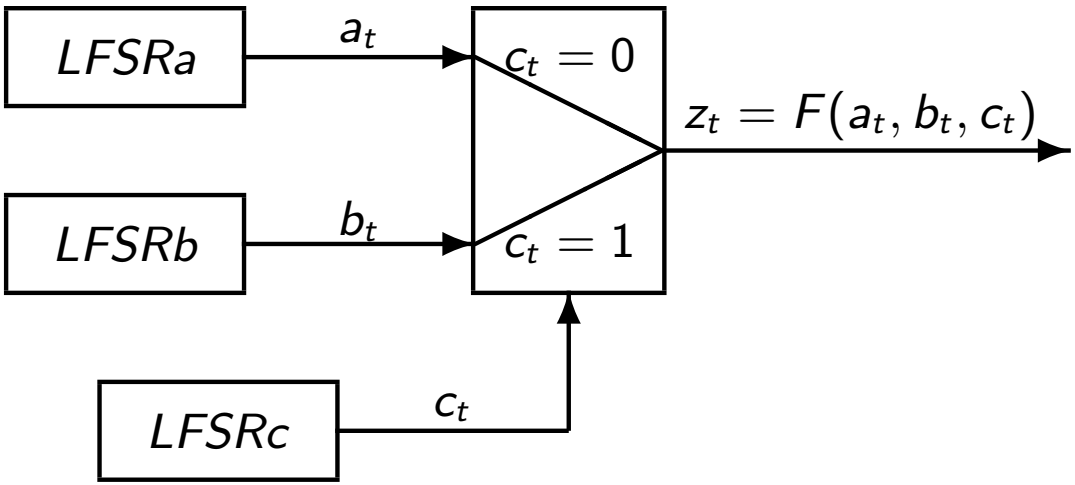
$F : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2$

$F(a_t, b_t, c_t) = a_t(1 + c_t) + b_t c_t$

a_t	b_t	c_t	z_t
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$z_t = a_t$ (75 % cases)

Example: Geffe generator



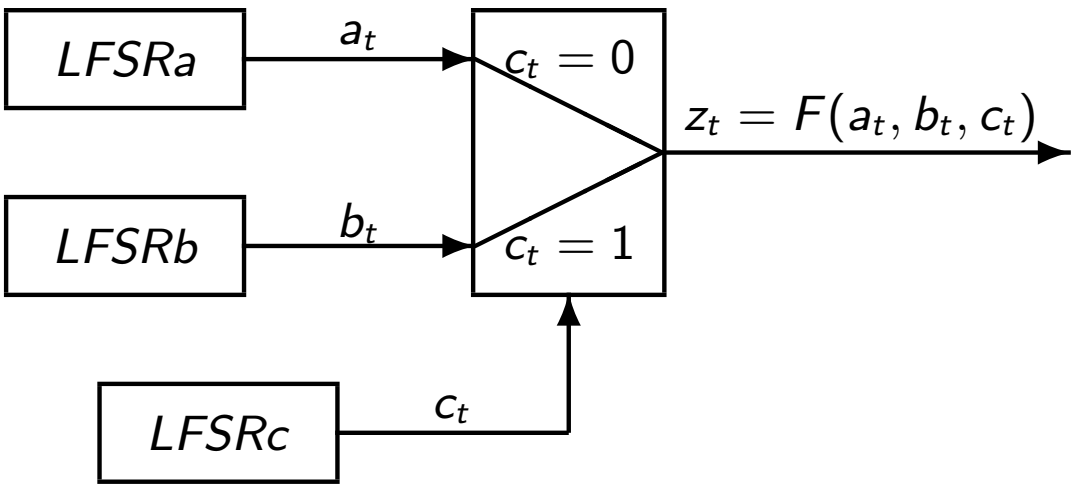
$$F : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2$$

$$F(a_t, b_t, c_t) = a_t(1 + c_t) + b_t c_t$$

a_t	b_t	c_t	z_t
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$z_t = a_t$ (75 % cases)

Example: Geffe generator



$$F : \mathbb{F}_2^3 \longrightarrow \mathbb{F}_2$$

$$F(a_t, b_t, c_t) = a_t(1 + c_t) + b_t c_t$$

a_t	b_t	c_t	z_t
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$z_t = a_t$ (75 % cases)

Attacks

- By brute force. Computational complexity $\mathcal{O}(2^k)$.
- Siegenthaler, "Correlation attacks", 1984.
- Meier and Staffelbach, "Fast correlation attacks", 1988.
  Complexity $\mathcal{O}(2^{ck})$, $(n/k = 100, t = 2 \text{ taps}, p = 0.75, c = 0.012)$.
- Gallager, "Low-Density Parity Check Codes", 1962.
- Johansson and Jönsson, "Improved fast correlation attack ... via convolutional codes", 1999.

Coding theory

$\mathbb{F}_q$ Galois field of q elements, consider $n, k \in \mathbb{N}$, $1 \leq k \leq n$.

An $[n, k]$-**code** over $\mathbb{F}_q$ is a k-dimensional linear subspace $\mathcal{C}$ of $\mathbb{F}_q^n$.

$G \in \mathbb{F}_q^{k \times n}$ is a **generator matrix** of $\mathcal{C}$ if $\operatorname{rank}(G) = k$ and

$$\mathcal{C} = \left\{ \boldsymbol{v} \in \mathbb{F}_q^n \mid \boldsymbol{v} = \boldsymbol{u}\, G,\ \boldsymbol{u} \in \mathbb{F}_q^k \right\}$$

$\boldsymbol{u} \in \mathbb{F}_q^k$ are **information words**, and $\boldsymbol{v} \in \mathcal{C}$ are **codewords**.

Since $\operatorname{rank}(G) = k$, there exists $H \in \mathbb{F}_q^{(n-k) \times n}$ with $\operatorname{rank}(H) = n - k$ and

$$\mathcal{C} = \left\{ \boldsymbol{v} \in \mathbb{F}_q^n \mid \boldsymbol{v} H^T = \boldsymbol{0} \right\}$$

H is called a **parity-check matrix** of $\mathcal{C}$.

If G is in **systematic form**

$$G = \begin{bmatrix} I_k & Q \end{bmatrix} \quad \longrightarrow \quad H = \begin{bmatrix} -Q^T & I_{n-k} \end{bmatrix}$$

Coding theory

$\mathbb{F}_q$ Galois field of q elements, consider $n, k \in \mathbb{N}$, $1 \leq k \leq n$.

An $[n, k]$-**code** over $\mathbb{F}_q$ is a k-dimensional linear subspace $\mathcal{C}$ of $\mathbb{F}_q^n$.

$G \in \mathbb{F}_q^{k \times n}$ is a **generator matrix** of $\mathcal{C}$ if $\operatorname{rank}(G) = k$ and

$$\mathcal{C} = \left\{ \boldsymbol{v} \in \mathbb{F}_q^n \mid \boldsymbol{v} = \boldsymbol{u}\, G,\ \boldsymbol{u} \in \mathbb{F}_q^k \right\}$$

$\boldsymbol{u} \in \mathbb{F}_q^k$ are **information words**, and $\boldsymbol{v} \in \mathcal{C}$ are **codewords**.

Since $\operatorname{rank}(G) = k$, there exists $H \in \mathbb{F}_q^{(n-k) \times n}$ with $\operatorname{rank}(H) = n - k$ and

$$\mathcal{C} = \left\{ \boldsymbol{v} \in \mathbb{F}_q^n \mid \boldsymbol{v} H^T = \boldsymbol{0} \right\}$$

H is called a **parity-check matrix** of $\mathcal{C}$.

If G is in **systematic form**

$$G = \begin{bmatrix} I_k & Q \end{bmatrix} \quad \longrightarrow \quad H = \begin{bmatrix} -Q^T & I_{n-k} \end{bmatrix}$$

Coding theory

$\mathbb{F}_q$ Galois field of q elements, consider $n, k \in \mathbb{N}$, $1 \leq k \leq n$.

An $[n, k]$-**code** over $\mathbb{F}_q$ is a k-dimensional linear subspace $\mathcal{C}$ of $\mathbb{F}_q^n$.

$G \in \mathbb{F}_q^{k \times n}$ is a **generator matrix** of $\mathcal{C}$ if $\operatorname{rank}(G) = k$ and

$$\mathcal{C} = \left\{ \boldsymbol{v} \in \mathbb{F}_q^n \mid \boldsymbol{v} = \boldsymbol{u}\, G,\ \boldsymbol{u} \in \mathbb{F}_q^k \right\}$$

$\boldsymbol{u} \in \mathbb{F}_q^k$ are **information words**, and $\boldsymbol{v} \in \mathcal{C}$ are **codewords**.

Since $\operatorname{rank}(G) = k$, there exists $H \in \mathbb{F}_q^{(n-k) \times n}$ with $\operatorname{rank}(H) = n - k$ and

$$\mathcal{C} = \left\{ \boldsymbol{v} \in \mathbb{F}_q^n \mid \boldsymbol{v} H^T = \boldsymbol{0} \right\}$$

H is called a **parity-check matrix** of $\mathcal{C}$.

If G is in **systematic form**

$$G = \begin{bmatrix} I_k & Q \end{bmatrix} \quad \longrightarrow \quad H = \begin{bmatrix} -Q^T & I_{n-k} \end{bmatrix}$$

A decoding problem

$$(k, f(x)) \text{ LFSR}, \quad f(x) = c_0 + c_1 x + \ldots, \quad + c_{k-1} x^{k-1} + x^k \in \mathbb{F}_2[x]$$

$$A = \begin{bmatrix} 0 & 0 & \ldots & 0 & c_0 \\ 1 & 0 & \ldots & 0 & c_1 \\ & & \ddots & & \\ 0 & 0 & \ldots & 0 & c_{k-2} \\ 0 & 0 & \ldots & 1 & c_{k-1} \end{bmatrix}, \qquad C = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{x}(0) = \begin{bmatrix} y_0 & y_1 & \ldots & y_{k-1} \end{bmatrix}, \quad x(i) = \begin{bmatrix} y_i & y_{i+1} & \ldots & y_{i+k-1} \end{bmatrix}$$

$$\left.\begin{array}{rcl} \boldsymbol{x}(i+1) & = & \boldsymbol{x}(i)\ A \\ \boldsymbol{y}(i) & = & \boldsymbol{x}(i)\ C \end{array}\right\} \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

$$(y_0, y_1, y_2, \ldots, y_n, \ldots) = (\boldsymbol{x}_0 C, \boldsymbol{x}_0 AC, \boldsymbol{x}_0 A^2 C, \ldots, \boldsymbol{x}_0 A^n C, \ldots)$$

A decoding problem

$$(k, f(x)) \text{ LFSR}, \quad f(x) = c_0 + c_1 x + \ldots, \quad + c_{k-1} x^{k-1} + x^k \in \mathbb{F}_2[x]$$

$$A = \begin{bmatrix} 0 & 0 & \ldots & 0 & c_0 \\ 1 & 0 & \ldots & 0 & c_1 \\ & & \ddots & & \\ 0 & 0 & \ldots & 0 & c_{k-2} \\ 0 & 0 & \ldots & 1 & c_{k-1} \end{bmatrix}, \qquad C = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{x}(0) = \begin{bmatrix} y_0 & y_1 & \ldots & y_{k-1} \end{bmatrix}, \quad x(i) = \begin{bmatrix} y_i & y_{i+1} & \ldots & y_{i+k-1} \end{bmatrix}$$

$$\left.\begin{array}{rcl} \boldsymbol{x}(i+1) & = & \boldsymbol{x}(i)\ A \\ \boldsymbol{y}(i) & = & \boldsymbol{x}(i)\ C \end{array}\right\} \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0$$

$$(y_0, y_1, y_2, \ldots, y_n, \ldots) = (\boldsymbol{x}_0 C, \boldsymbol{x}_0 AC, \boldsymbol{x}_0 A^2 C, \ldots, \boldsymbol{x}_0 A^n C, \ldots)$$

Assume we do not know $\boldsymbol{y}$, $\boldsymbol{x}_0$. Assume we know

$$\boldsymbol{z} = (z_0, z_1, z_2, \ldots, z_{k-1}, z_k, z_{k+1}, \ldots, z_{n-1})$$

correlated to $\boldsymbol{y}$ with probability p (usually $0.5 \leq p \leq 0.75$).

$\boldsymbol{z}$ can be viewed as a perturbation of the sequence $\boldsymbol{y}$ by a binary symmetric memoryless noise channel with

$$p = Pr(z_t = y_t) = 1 - \varepsilon$$

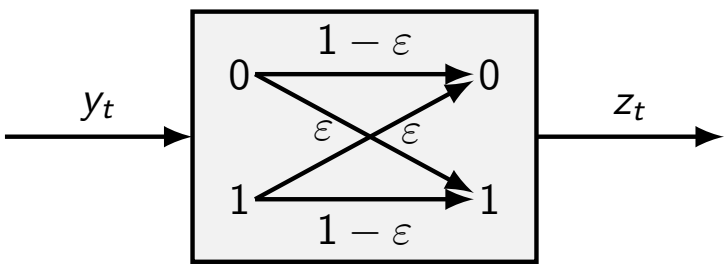
Assume we do not know $\boldsymbol{y}$, $\boldsymbol{x}_0$. Assume we know

$$\boldsymbol{z} = (z_0, z_1, z_2, \ldots, z_{k-1}, z_k, z_{k+1}, \ldots, z_{n-1})$$

correlated to $\boldsymbol{y}$ with probability p (usually $0.5 \leq p \leq 0.75$).

$\boldsymbol{z}$ can be viewed as a perturbation of the sequence $\boldsymbol{y}$ by a binary symmetric memoryless noise channel with

$$p = Pr(z_t = y_t) = 1 - \varepsilon$$



From now on $n = mk$ for some m such that $k < n < 2^k - 1$. Then

$$\begin{bmatrix} C & AC & A^2C & A^3C & \cdots & A^{k-1}C \end{bmatrix} = I_k,$$

consequently

$$\begin{bmatrix} A^kC & A^{k+1}C & A^{k+2}C & A^{k+3}C & \cdots & A^{2k-1}C \end{bmatrix}$$
$$= A^k \begin{bmatrix} C & AC & A^2 & A^3 & \cdots & A^{k-1}C \end{bmatrix} = A^k,$$

denoting by

$$G = \begin{bmatrix} I_k & A^k & A^{2k} & A^{3k} & \cdots & A^{(m-2)k} & A^{(m-1)k} \end{bmatrix},$$

the sequence $\boldsymbol{y}$ is a codeword of the code $\mathcal{C}$ generated by G

$$\boldsymbol{y} = \boldsymbol{x}_0 G,$$

and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, $\boldsymbol{e} \in \mathbb{F}_2^n$ with average Hamming weight $\mathrm{wt}(\boldsymbol{e}) = \lfloor \varepsilon n \rfloor$.

Restatement of the problem

Given a received word $\boldsymbol{z}$, find the transmitted codeword $\boldsymbol{y}$.

From now on $n = mk$ for some m such that $k < n < 2^k - 1$. Then

$$\begin{bmatrix} C & AC & A^2C & A^3C & \cdots & A^{k-1}C \end{bmatrix} = I_k,$$

consequently

$$\begin{bmatrix} A^kC & A^{k+1}C & A^{k+2}C & A^{k+3}C & \cdots & A^{2k-1}C \end{bmatrix}$$
$$= A^k \begin{bmatrix} C & AC & A^2 & A^3 & \cdots & A^{k-1}C \end{bmatrix} = A^k,$$

denoting by

$$G = \begin{bmatrix} I_k & A^k & A^{2k} & A^{3k} & \cdots & A^{(m-2)k} & A^{(m-1)k} \end{bmatrix},$$

the sequence $\boldsymbol{y}$ is a codeword of the code $\mathcal{C}$ generated by G

$$\boldsymbol{y} = \boldsymbol{x}_0 G,$$

and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, $\boldsymbol{e} \in \mathbb{F}_2^n$ with average Hamming weight $\mathrm{wt}(\boldsymbol{e}) = \lfloor \varepsilon n \rfloor$.

Restatement of the problem

Given a received word $\boldsymbol{z}$, find the transmitted codeword $\boldsymbol{y}$.

From now on $n = mk$ for some m such that $k < n < 2^k - 1$. Then

$$\begin{bmatrix} C & AC & A^2C & A^3C & \cdots & A^{k-1}C \end{bmatrix} = I_k,$$

consequently

$$\begin{bmatrix} A^kC & A^{k+1}C & A^{k+2}C & A^{k+3}C & \cdots & A^{2k-1}C \end{bmatrix}$$
$$= A^k \begin{bmatrix} C & AC & A^2 & A^3 & \cdots & A^{k-1}C \end{bmatrix} = A^k,$$

denoting by

$$G = \begin{bmatrix} I_k & A^k & A^{2k} & A^{3k} & \cdots & A^{(m-2)k} & A^{(m-1)k} \end{bmatrix},$$

the sequence $\boldsymbol{y}$ is a codeword of the code $\mathcal{C}$ generated by G

$$\boldsymbol{y} = \boldsymbol{x}_0 G,$$

and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, $\boldsymbol{e} \in \mathbb{F}_2^n$ with average Hamming weight $\mathrm{wt}(\boldsymbol{e}) = \lfloor \varepsilon n \rfloor$.

Restatement of the problem

Given a received word $\boldsymbol{z}$, find the transmitted codeword $\boldsymbol{y}$.

From now on $n = mk$ for some m such that $k < n < 2^k - 1$. Then

$$\begin{bmatrix} C & AC & A^2C & A^3C & \cdots & A^{k-1}C \end{bmatrix} = I_k,$$

consequently

$$\begin{bmatrix} A^kC & A^{k+1}C & A^{k+2}C & A^{k+3}C & \cdots & A^{2k-1}C \end{bmatrix}$$
$$= A^k \begin{bmatrix} C & AC & A^2 & A^3 & \cdots & A^{k-1}C \end{bmatrix} = A^k,$$

denoting by

$$G = \begin{bmatrix} I_k & A^k & A^{2k} & A^{3k} & \cdots & A^{(m-2)k} & A^{(m-1)k} \end{bmatrix},$$

the sequence $\boldsymbol{y}$ is a codeword of the code $\mathcal{C}$ generated by G

$$\boldsymbol{y} = \boldsymbol{x}_0 G,$$

and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, $\boldsymbol{e} \in \mathbb{F}_2^n$ with average Hamming weight $\mathrm{wt}(\boldsymbol{e}) = \lfloor \varepsilon n \rfloor$.

Restatement of the problem

Given a received word $\boldsymbol{z}$, find the transmitted codeword $\boldsymbol{y}$.

G in systematic form, then denoting $B = (A^k)^T$

$$H = \begin{bmatrix} B & I_k & & \\ B^2 & & I_k & \\ \vdots & & & \ddots \\ B^{m-1} & & & I_k \end{bmatrix}$$

Notice that

$$zH^T = eH^T = s \neq 0, \quad s \textbf{ syndrome}.$$

We take

$$y = (y_0, y_1, y_2, \ldots, y_{m-1}), \quad z = (z_0, z_1, z_2, \ldots, z_{m-1}),$$
$$y_i = (y_{ik}, y_{ik+1}, y_{ik+2}, \ldots, y_{(i+1)k-1}), \quad (y_0 = x_0)$$
$$z_i = (z_{ik}, z_{ik+1}, z_{ik+2}, \ldots, z_{(i+1)k-1}),$$

$$e = (e_0, e_1, e_2, \ldots, e_{m-1}), \quad s = (s_0, s_1, s_2, \ldots, s_{m-1}),$$
$$e_i = (e_{ik}, e_{ik+1}, e_{ik+2}, \ldots, e_{(i+1)k-1}),$$
$$s_i = (s_{ik}, s_{ik+1}, s_{ik+2}, \ldots, s_{(i+1)k-1}),$$

G in systematic form, then denoting $B = (A^k)^T$

$$H = \begin{bmatrix} B & I_k & & \\ B^2 & & I_k & \\ \vdots & & & \ddots \\ B^{m-1} & & & I_k \end{bmatrix}$$

Notice that

$$\boldsymbol{z}H^T = \boldsymbol{e}H^T = \boldsymbol{s} \neq 0, \quad s \textbf{ syndrome}.$$

We take

$$\boldsymbol{y} = (\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{m-1}), \quad \boldsymbol{z} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{m-1}),$$
$$\boldsymbol{y}_i = (y_{ik}, y_{ik+1}, y_{ik+2}, \ldots, y_{(i+1)k-1}), \quad (\boldsymbol{y}_0 = x_0)$$
$$\boldsymbol{z}_i = (z_{ik}, z_{ik+1}, z_{ik+2}, \ldots, z_{(i+1)k-1}),$$

$$\boldsymbol{e} = (\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{m-1}), \quad \boldsymbol{s} = (\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{m-1}),$$
$$\boldsymbol{e}_i = (e_{ik}, e_{ik+1}, e_{ik+2}, \ldots, e_{(i+1)k-1}),$$
$$\boldsymbol{s}_i = (s_{ik}, s_{ik+1}, s_{ik+2}, \ldots, s_{(i+1)k-1}),$$

G in systematic form, then denoting $B = (A^k)^T$

$$H = \begin{bmatrix} B & I_k & & \\ B^2 & & I_k & \\ \vdots & & & \ddots \\ B^{m-1} & & & I_k \end{bmatrix}$$

Notice that

$$\boldsymbol{z}H^T = \boldsymbol{e}H^T = \boldsymbol{s} \neq 0, \quad s \textbf{ syndrome}.$$

We take

$$\boldsymbol{y} = (\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{m-1}), \quad \boldsymbol{z} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{m-1}),$$
$$\boldsymbol{y}_i = (y_{ik}, y_{ik+1}, y_{ik+2}, \ldots, y_{(i+1)k-1}), \quad (\boldsymbol{y}_0 = x_0)$$
$$\boldsymbol{z}_i = (z_{ik}, z_{ik+1}, z_{ik+2}, \ldots, z_{(i+1)k-1}),$$

$$\boldsymbol{e} = (\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{m-1}), \quad s = (\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{m-1}),$$
$$\boldsymbol{e}_i = (e_{ik}, e_{ik+1}, e_{ik+2}, \ldots, e_{(i+1)k-1}),$$
$$\boldsymbol{s}_i = (s_{ik}, s_{ik+1}, s_{ik+2}, \ldots, s_{(i+1)k-1}),$$

- Decoding algorithm 1

$$\left[\begin{array}{c|ccc} B & I_k & & \\ B^2 & & I_k & \\ \vdots & & & \ddots \\ B^{m-1} & & & I_k \end{array}\right] = \begin{bmatrix} \tilde{B} & I_{(m-1)k} \end{bmatrix}$$

$$e = (e_0, e_1, e_2, \ldots, e_{m-1}) = (e_0, \tilde{e}), \quad \tilde{e} = (e_1, \ldots, e_{m-1})$$
$$zH^T = s \quad \longleftrightarrow \quad eH^T = s \quad \longleftrightarrow \quad e_0\tilde{B}^T + \tilde{e} = s$$

- Decoding algorithm 2

$$\left[\begin{array}{cc|c} B & I_k & \\ \hline B^2 & & I_k \\ \vdots & & \ddots \\ B^{m-1} & & I_k \end{array}\right] = \begin{bmatrix} H_1 & 0 \\ H_2 & I_{(m-2)k} \end{bmatrix} = \left[\begin{array}{cc|c} B & I_k & 0 \\ \hline B_1 & 0 & I_{(m-2)k} \end{array}\right]$$

$$e = (\tilde{e}_0, \tilde{e}_1) = (e_0, e_1, \tilde{e}_1), \tilde{e}_1 = (e_2, \ldots, e_{m-1}),\ s = (s_0, \tilde{s}_1),\ \tilde{s}_1 = (s_1, \ldots, \tilde{s}_{m-2})$$

$$zH^T = s \leftrightarrow eH^T = \begin{bmatrix} \tilde{e}_0 H_1^T & \tilde{e}_0 H_2^T + \tilde{e}_1 \end{bmatrix} = \begin{bmatrix} s_0 & \tilde{s}_1 \end{bmatrix} \leftrightarrow \begin{cases} \tilde{e}_0 H_1^T = s_0 \\ e_0 B_1^T + \tilde{e}_1 = \tilde{s}_1 \end{cases}$$

Decoding algorithm 1

$$\boldsymbol{e} \in \mathbb{F}_2^n \ ? \qquad \boldsymbol{e}H^T = \boldsymbol{s}, \quad \mathrm{wt}(\boldsymbol{e}) = w$$

Decoding algorithm 1

$\boldsymbol{e} \in \mathbb{F}_2^n$? $\quad \boldsymbol{e}H^T = \boldsymbol{s}, \quad \mathrm{wt}(\boldsymbol{e}) = w$

Algorithm 1

input : H, $\boldsymbol{s}$, and ω
output: $\boldsymbol{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{e}H^T = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{e}) = \omega$

1 **for** $\boldsymbol{e}_0 \in \mathbb{F}_2^k$ *such that* $\mathrm{wt}(\boldsymbol{e}_0) = \lfloor \varepsilon k \rfloor$ **do**
2 compute $\tilde{\boldsymbol{e}}$ as $\tilde{\boldsymbol{e}} = \boldsymbol{e}_0 \tilde{B}^T + \boldsymbol{s}$
3 let $\boldsymbol{e} = (\boldsymbol{e}_0, \tilde{\boldsymbol{e}})$
4 **if** $\mathrm{wt}(\boldsymbol{e}) = \omega$ **then**
5 STOP
6 **end**
7 **end**

Decoding algorithm 1

$\boldsymbol{e} \in \mathbb{F}_2^n$? $\quad \boldsymbol{e}H^T = \boldsymbol{s}, \quad \mathrm{wt}(\boldsymbol{e}) = w$

Algorithm 1

input : H, $\boldsymbol{s}$, and ω
output: $\boldsymbol{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{e}H^T = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{e}) = \omega$

1 **for** $\boldsymbol{e}_0 \in \mathbb{F}_2^k$ *such that* $\mathrm{wt}(\boldsymbol{e}_0) = \lfloor \varepsilon k \rfloor$ **do**
2 compute $\tilde{\boldsymbol{e}}$ as $\tilde{\boldsymbol{e}} = \boldsymbol{e}_0 \tilde{B}^T + \boldsymbol{s}$
3 let $\boldsymbol{e} = (\boldsymbol{e}_0, \tilde{\boldsymbol{e}})$
4 **if** $\mathrm{wt}(\boldsymbol{e}) = \omega$ **then**
5 STOP
6 **end**
7 **end**

Right vector $\boldsymbol{e}$ in 50 % of cases.

Decoding algorithm 1

$\boldsymbol{e} \in \mathbb{F}_2^n$? $\quad \boldsymbol{e}H^T = \boldsymbol{s}, \quad \mathrm{wt}(\boldsymbol{e}) = w$

Algorithm 1

input : H, $\boldsymbol{s}$, and ω
output: $\boldsymbol{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{e}H^T = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{e}) = \omega$

1 **for** $\boldsymbol{e}_0 \in \mathbb{F}_2^k$ *such that* $\lfloor \varepsilon k \rfloor - \delta \leq \mathrm{wt}(\boldsymbol{e}_0) \leq \lfloor \varepsilon k \rfloor + \delta$ **do**
2 compute $\tilde{\boldsymbol{e}}$ as $\tilde{\boldsymbol{e}} = \boldsymbol{e}_0 \tilde{B}^T + \boldsymbol{s}$
3 let $\boldsymbol{e} = (\boldsymbol{e}_0, \tilde{\boldsymbol{e}})$
4 **if** $\mathrm{wt}(\boldsymbol{e}) = \omega$ **then**
5 STOP
6 **end**
7 **end**

Right vector $\boldsymbol{e}$ in 100 % of cases.

Decoding algorithm 2

- Based on a method by Becker, Joux, May and Meurer [2].

Decoding algorithm 2

- Based on a method by Becker, Joux, May and Meurer [2].
- Attention focused on efficiently computing $\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k}$:

$$\boldsymbol{z}H^T = s \longleftrightarrow \begin{cases} \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0 \\ \boldsymbol{e}_0 B_1^T + \tilde{\boldsymbol{e}}_1 = \tilde{\boldsymbol{s}}_1 \end{cases} \qquad \boldsymbol{e} = (\tilde{\boldsymbol{e}}_0, \tilde{\boldsymbol{e}}_1) \quad \boldsymbol{s} = (\tilde{\boldsymbol{s}}_0, \tilde{\boldsymbol{s}}_1)$$

Decoding algorithm 2

- Based on a method by Becker, Joux, May and Meurer [2].
- Attention focused on efficiently computing $\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k}$:

$$\boldsymbol{z}H^T = s \longleftrightarrow \begin{cases} \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0 \\ \boldsymbol{e}_0 B_1^T + \tilde{\boldsymbol{e}}_1 = \tilde{\boldsymbol{s}}_1 \end{cases} \qquad \boldsymbol{e} = (\tilde{\boldsymbol{e}}_0, \tilde{\boldsymbol{e}}_1) \quad \boldsymbol{s} = (\tilde{\boldsymbol{s}}_0, \tilde{\boldsymbol{s}}_1)$$

- Additional hypotheses on the weight distribution of $\boldsymbol{e} \in \mathbb{F}_2^n$ required.

Decoding algorithm 2

- Based on a method by Becker, Joux, May and Meurer [2].
- Attention focused on efficiently computing $\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k}$:

$$\boldsymbol{z}H^T = s \longleftrightarrow \begin{cases} \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0 \\ \boldsymbol{e}_0 B_1^T + \tilde{\boldsymbol{e}}_1 = \tilde{\boldsymbol{s}}_1 \end{cases} \qquad \boldsymbol{e} = (\tilde{\boldsymbol{e}}_0, \tilde{\boldsymbol{e}}_1) \quad \boldsymbol{s} = (\tilde{\boldsymbol{s}}_0, \tilde{\boldsymbol{s}}_1)$$

- Additional hypotheses on the weight distribution of $\boldsymbol{e} \in \mathbb{F}_2^n$ required.
- Given $\mathcal{L}_1, \mathcal{L}_2 \subseteq \mathbb{F}_2^{2k}$, $\boldsymbol{t} \in \mathbb{F}_2^r$, we use the **Merge-Join algorithm** of [2] (based on a classical matching algorithm by Knuth [6]) to obtain

$$\mathcal{L} = \Big\{ \boldsymbol{x} + \boldsymbol{y} \in \mathbb{F}_2^{2k} \mid \boldsymbol{x} \in \mathcal{L}_1, \boldsymbol{y} \in \mathcal{L}_2, \\ \mathrm{wt}(\boldsymbol{x} + \boldsymbol{y}) = p \quad \text{and} \quad \left((\boldsymbol{x} + \boldsymbol{y}) H_1^T\right)[k - r + 1 : k] = \boldsymbol{t} \Big\}.$$

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

1 Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

1 Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;
2 Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

1 Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;
2 Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;
3 Choose random vectors $\boldsymbol{v}_1 \in \mathbb{F}_2^{r_1}$, and $\boldsymbol{u}_1, \boldsymbol{u}_3 \in \mathbb{F}_2^{r_2}$;
4 Define vectors $\boldsymbol{v}_2 = \boldsymbol{s}_0[k - r_1 + 1 : k] + \boldsymbol{v}_1$,

$\boldsymbol{u}_2 = \boldsymbol{v}_1[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_1, \quad \boldsymbol{u}_4 = \boldsymbol{v}_2[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_3$

▸ Vectors

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$

output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

1 Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;

2 Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;

3 Choose random vectors $\boldsymbol{v}_1 \in \mathbb{F}_2^{r_1}$, and $\boldsymbol{u}_1, \boldsymbol{u}_3 \in \mathbb{F}_2^{r_2}$;

4 Define vectors $\boldsymbol{v}_2 = \boldsymbol{s}_0[k - r_1 + 1 : k] + \boldsymbol{v}_1$,

$\boldsymbol{u}_2 = \boldsymbol{v}_1[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_1, \quad \boldsymbol{u}_4 = \boldsymbol{v}_2[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_3$ ▸ Vectors

5 **for** $i = 1, 2, 3, 4$ **do**

6 Choose random partitions $\mathbb{P}_i = \{\mathcal{P}_{2i-1}, \mathcal{P}_{2i}\}$ of $\{1, 2, \ldots, 2k\}$, $|\mathcal{P}_{2i-1}| = |\mathcal{P}_{2i}|$

▸ Partitions and basic sets

7 Obtain the basic sets $\mathcal{L}_{2i-1}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i}\right\}$

$\mathcal{L}_{2i}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i-1}\right\}$

8 **end**

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$

output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;

Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;

Choose random vectors $\boldsymbol{v}_1 \in \mathbb{F}_2^{r_1}$, and $\boldsymbol{u}_1, \boldsymbol{u}_3 \in \mathbb{F}_2^{r_2}$;

Define vectors $\boldsymbol{v}_2 = \boldsymbol{s}_0[k - r_1 + 1 : k] + \boldsymbol{v}_1$,

$\boldsymbol{u}_2 = \boldsymbol{v}_1[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_1$, $\boldsymbol{u}_4 = \boldsymbol{v}_2[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_3$ ▸ Vectors

for $i = 1, 2, 3, 4$ **do**

  Choose random partitions $\mathbb{P}_i = \{\mathcal{P}_{2i-1}, \mathcal{P}_{2i}\}$ of $\{1, 2, \ldots, 2k\}$, $|\mathcal{P}_{2i-1}| = |\mathcal{P}_{2i}|$

  ▸ Partitions and basic sets

  Obtain the basic sets $\mathcal{L}_{2i-1}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i}\right\}$

  $\mathcal{L}_{2i}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i-1}\right\}$

end

for $i = 1, 2, 3, 4$ **do**

  use $\left(\mathcal{L}_{2i-1}^{(3)}, \mathcal{L}_{2i}^{(3)}\right)$, $\boldsymbol{u}_i$, and the Merge-Join algorithm to obtain the sets $\mathcal{L}_i^{(2)}$

end

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;
Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;
Choose random vectors $\boldsymbol{v}_1 \in \mathbb{F}_2^{r_1}$, and $\boldsymbol{u}_1, \boldsymbol{u}_3 \in \mathbb{F}_2^{r_2}$;
Define vectors $\boldsymbol{v}_2 = \boldsymbol{s}_0[k - r_1 + 1 : k] + \boldsymbol{v}_1$,

$\boldsymbol{u}_2 = \boldsymbol{v}_1[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_1, \quad \boldsymbol{u}_4 = \boldsymbol{v}_2[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_3$ ▸ Vectors

for $i = 1, 2, 3, 4$ **do**
  Choose random partitions $\mathbb{P}_i = \{\mathcal{P}_{2i-1}, \mathcal{P}_{2i}\}$ of $\{1, 2, \ldots, 2k\}$, $|\mathcal{P}_{2i-1}| = |\mathcal{P}_{2i}|$
  ▸ Partitions and basic sets
  Obtain the basic sets $\mathcal{L}_{2i-1}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i}\right\}$
  $\mathcal{L}_{2i}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i-1}\right\}$
end
for $i = 1, 2, 3, 4$ **do**
  use $\left(\mathcal{L}_{2i-1}^{(3)}, \mathcal{L}_{2i}^{(3)}\right)$, $\boldsymbol{u}_i$, and the Merge-Join algorithm to obtain the sets $\mathcal{L}_i^{(2)}$
end
for $j = 1, 2$ **do**
  use $\left(\mathcal{L}_{2j-1}^{(2)}, \mathcal{L}_{2j}^{(2)}\right)$, $\boldsymbol{v}_j$, and the Merge-Join algorithm to obtain the set $\mathcal{L}_j^{(1)}$
end

Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;
Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;
Choose random vectors $\boldsymbol{v}_1 \in \mathbb{F}_2^{r_1}$, and $\boldsymbol{u}_1, \boldsymbol{u}_3 \in \mathbb{F}_2^{r_2}$;
Define vectors $\boldsymbol{v}_2 = \boldsymbol{s}_0[k - r_1 + 1 : k] + \boldsymbol{v}_1$,

$\boldsymbol{u}_2 = \boldsymbol{v}_1[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_1, \quad \boldsymbol{u}_4 = \boldsymbol{v}_2[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_3$ ▸ Vectors

for $i = 1, 2, 3, 4$ **do**
  Choose random partitions $\mathbb{P}_i = \{\mathcal{P}_{2i-1}, \mathcal{P}_{2i}\}$ of $\{1, 2, \ldots, 2k\}$, $|\mathcal{P}_{2i-1}| = |\mathcal{P}_{2i}|$
  ▸ Partitions and basic sets
  Obtain the basic sets $\mathcal{L}_{2i-1}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i}\right\}$
  $\mathcal{L}_{2i}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i-1}\right\}$
end
for $i = 1, 2, 3, 4$ **do**
  use $\left(\mathcal{L}_{2i-1}^{(3)}, \mathcal{L}_{2i}^{(3)}\right)$, $\boldsymbol{u}_i$, and the Merge-Join algorithm to obtain the sets $\mathcal{L}_i^{(2)}$
end
for $j = 1, 2$ **do**
  use $\left(\mathcal{L}_{2j-1}^{(2)}, \mathcal{L}_{2j}^{(2)}\right)$, $\boldsymbol{v}_j$, and the Merge-Join algorithm to obtain the set $\mathcal{L}_j^{(1)}$
end
Use $\left(\mathcal{L}_1^{(1)}, \mathcal{L}_2^{(1)}\right)$, $\boldsymbol{s}_0$, and the Merge-Join algorithm to obtain the set $\mathcal{L}$ ▸ $\mathcal{L}$ set

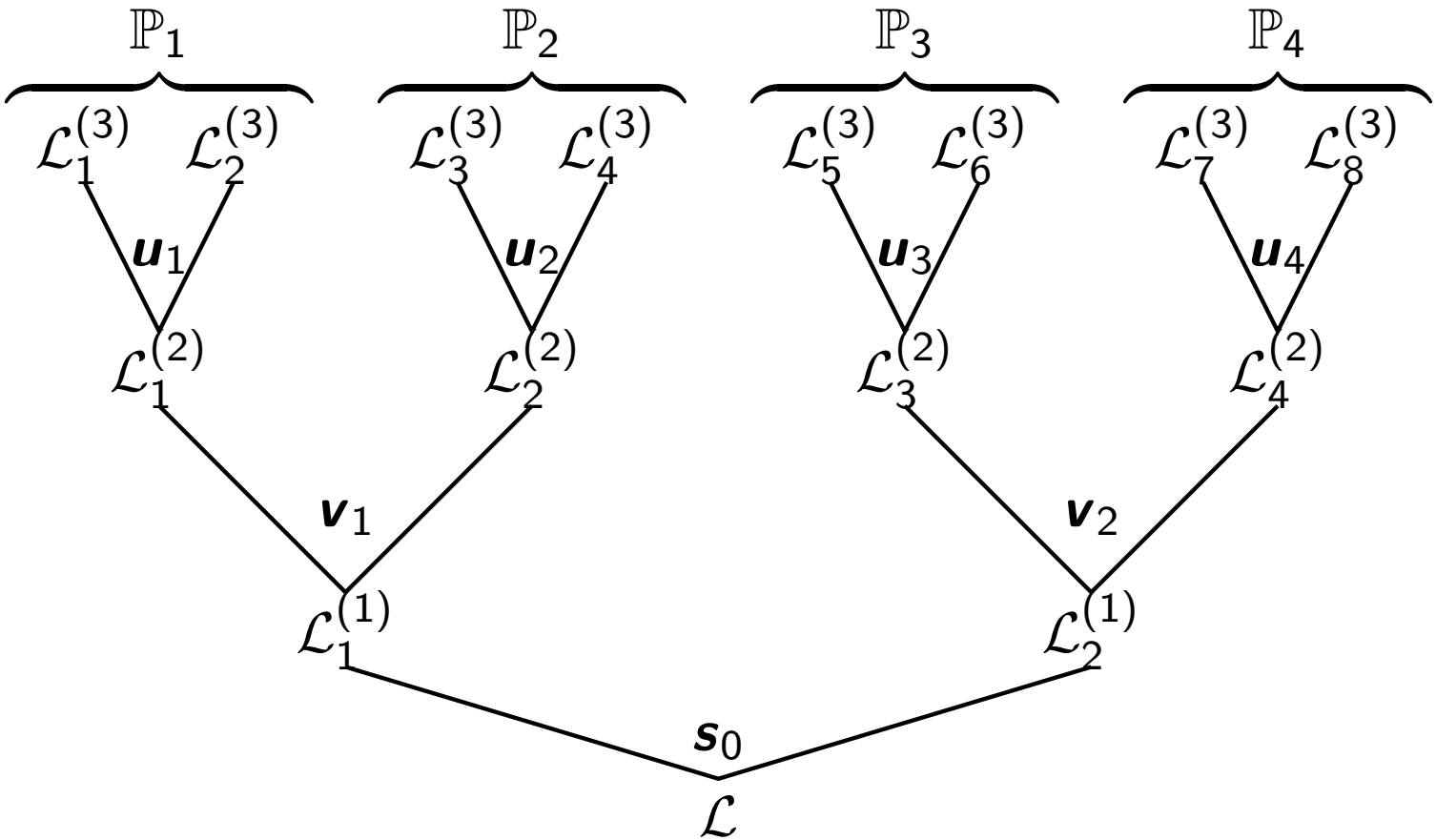
Algorithm 2

input : H_1, $\boldsymbol{s}_0$, and p with $1 \leq p \leq 2k$
output: $\mathcal{L} = \{\tilde{\boldsymbol{e}}_0 \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p \text{ and } \tilde{\boldsymbol{e}}_0 H_1^T = \boldsymbol{s}_0\}$

Define $p_1 = p/2 + \varepsilon_1$ and $p_2 = p_1/2 + \varepsilon_2$ such that p_1 and p_2 are even numbers;
Choose random integers r_1, r_2 such that $1 \leq r_2 \leq r_1 \leq k$;
Choose random vectors $\boldsymbol{v}_1 \in \mathbb{F}_2^{r_1}$, and $\boldsymbol{u}_1, \boldsymbol{u}_3 \in \mathbb{F}_2^{r_2}$;
Define vectors $\boldsymbol{v}_2 = \boldsymbol{s}_0[k - r_1 + 1 : k] + \boldsymbol{v}_1$,

$\boldsymbol{u}_2 = \boldsymbol{v}_1[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_1, \quad \boldsymbol{u}_4 = \boldsymbol{v}_2[r_1 - r_2 + 1 : r_1] + \boldsymbol{u}_3$ ▸ Vectors

for $i = 1, 2, 3, 4$ **do**
  Choose random partitions $\mathbb{P}_i = \{\mathcal{P}_{2i-1}, \mathcal{P}_{2i}\}$ of $\{1, 2, \ldots, 2k\}$, $|\mathcal{P}_{2i-1}| = |\mathcal{P}_{2i}|$
  ▸ Partitions and basic sets
  Obtain the basic sets $\mathcal{L}_{2i-1}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i}\right\}$
  $\mathcal{L}_{2i}^{(3)} = \left\{\boldsymbol{a} \in \mathbb{F}_2^{2k} \mid \mathrm{wt}(\boldsymbol{a}) = p_2/2 \text{ and } a_l = 0 \text{ for all } l \in \mathcal{P}_{2i-1}\right\}$
end
for $i = 1, 2, 3, 4$ **do**
  use $\left(\mathcal{L}_{2i-1}^{(3)}, \mathcal{L}_{2i}^{(3)}\right)$, $\boldsymbol{u}_i$, and the Merge-Join algorithm to obtain the sets $\mathcal{L}_i^{(2)}$
end
for $j = 1, 2$ **do**
  use $\left(\mathcal{L}_{2j-1}^{(2)}, \mathcal{L}_{2j}^{(2)}\right)$, $\boldsymbol{v}_j$, and the Merge-Join algorithm to obtain the set $\mathcal{L}_j^{(1)}$
end
Use $\left(\mathcal{L}_1^{(1)}, \mathcal{L}_2^{(1)}\right)$, $\boldsymbol{s}_0$, and the Merge-Join algorithm to obtain the set $\mathcal{L}$ ▸ $\mathcal{L}$ set



$$\boldsymbol{e} = \boldsymbol{d}_1 + \boldsymbol{d}_2 = (\boldsymbol{c}_1^{(1)} + \boldsymbol{c}_2^{(1)}) + (\boldsymbol{c}_1^{(2)} + \boldsymbol{c}_2^{(2)})$$
$$= (\boldsymbol{b}_1^{(1)} + \boldsymbol{b}_2^{(1)}) + (\boldsymbol{b}_1^{(2)} + \boldsymbol{b}_2^{(2)}) + (\boldsymbol{b}_1^{(3)} + \boldsymbol{b}_2^{(3)}) + (\boldsymbol{b}_1^{(4)} + \boldsymbol{b}_2^{(4)})$$

Algorithm 3

input : H, $\boldsymbol{s}$, $\mathcal{L}$, ω
output: $\boldsymbol{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{e}H^T = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{e}) = \omega$

for $\tilde{\boldsymbol{e}}_0 \in \mathcal{L}$ **do** % $\mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p$
  Compute $\tilde{\boldsymbol{e}}_1 \in \mathbb{F}_2^{n-2k}$ as $\tilde{\boldsymbol{e}}_1 = \tilde{\boldsymbol{e}}_0 B_1^T + \tilde{\boldsymbol{s}}_1$;
  if $\mathrm{wt}(\tilde{\boldsymbol{e}}_1) = \omega - p$ **then**
    define $\boldsymbol{e} = (\tilde{\boldsymbol{e}}_0, \tilde{\boldsymbol{e}}_1)$
  end
end

Algorithm 3

input : H, $\boldsymbol{s}$, $\mathcal{L}$, ω
output: $\boldsymbol{e} \in \mathbb{F}_2^n$ such that $\boldsymbol{e}H^T = \boldsymbol{s}$ and $\mathrm{wt}(\boldsymbol{e}) = \omega$

1 **for** $\tilde{\boldsymbol{e}}_0 \in \mathcal{L}$ **do** % $\mathrm{wt}(\tilde{\boldsymbol{e}}_0) = p$
2 Compute $\tilde{\boldsymbol{e}}_1 \in \mathbb{F}_2^{n-2k}$ as $\tilde{\boldsymbol{e}}_1 = \tilde{\boldsymbol{e}}_0 B_1^T + \tilde{\boldsymbol{s}}_1$;
3 **if** $\mathrm{wt}(\tilde{\boldsymbol{e}}_1) = \omega - p$ **then**
4 define $\boldsymbol{e} = (\tilde{\boldsymbol{e}}_0, \tilde{\boldsymbol{e}}_1)$
5 **end**
6 **end**

Using
Right vector $\boldsymbol{e}$ in 100 % of cases.

References I

[1] Martin Ågren, Carl Löndahl, Martin Hell, and Thomas Johansson.
A survey on fast correlation attacks.
Cryptography and Communications, 4(3–4):173–202, 2012.

[2] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer.
Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding.
In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer-Verlag, Berlin, 2012.

[3] Anne Canteaut and María Naya-Plasencia.
Correlation attacks on combination generators.
Cryptography and Communications, 4(3–4):147–171, 2012.

[4] Vladimir V. Chepyzhov, Thomas Johansson, and Ben Smeets.
A simple algorithm for fast correlation attacks on stream ciphers.
In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 181–195. Springer-Verlag, Berlin, 2001.

References II

[5] Thomas Johansson and Fredrik Jönsson.
Theoretical analysis of a correlation attack based on convolutional codes.
IEEE Transactions on Information Theory, 48(8):2173–2181, 2002.

[6] Donald E. Knuth.
The Art of Computer Programming. Sorting and Searching, volume 3.
Addison-Wesley, Boston, MA, 1998.

[7] Peizhong Lu and Lianzhen Huang.
A new correlation attack on LFSR sequences with high error tolerance.
Progress in Computer Science and Applied Logic, 23:67–83, 2004.

[8] Willi Meier.
Fast correlation attacks: methods and countermeasures.
In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 55–67. Springer-Verlag, Berlin, 2011.

[9] Willi Meier and Othmar Staffelbach.
Fast correlation attacks on stream ciphers.
In Christoph G. Günter, editor, *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314.
Springer-Verlag, Berlin, 1988.

References III

[10] Willi Meier and Othmar Staffelbach.
Fast correlation attacks on certain stream ciphers.
Journal of Cryptology, 1(3):159–176, 1989.

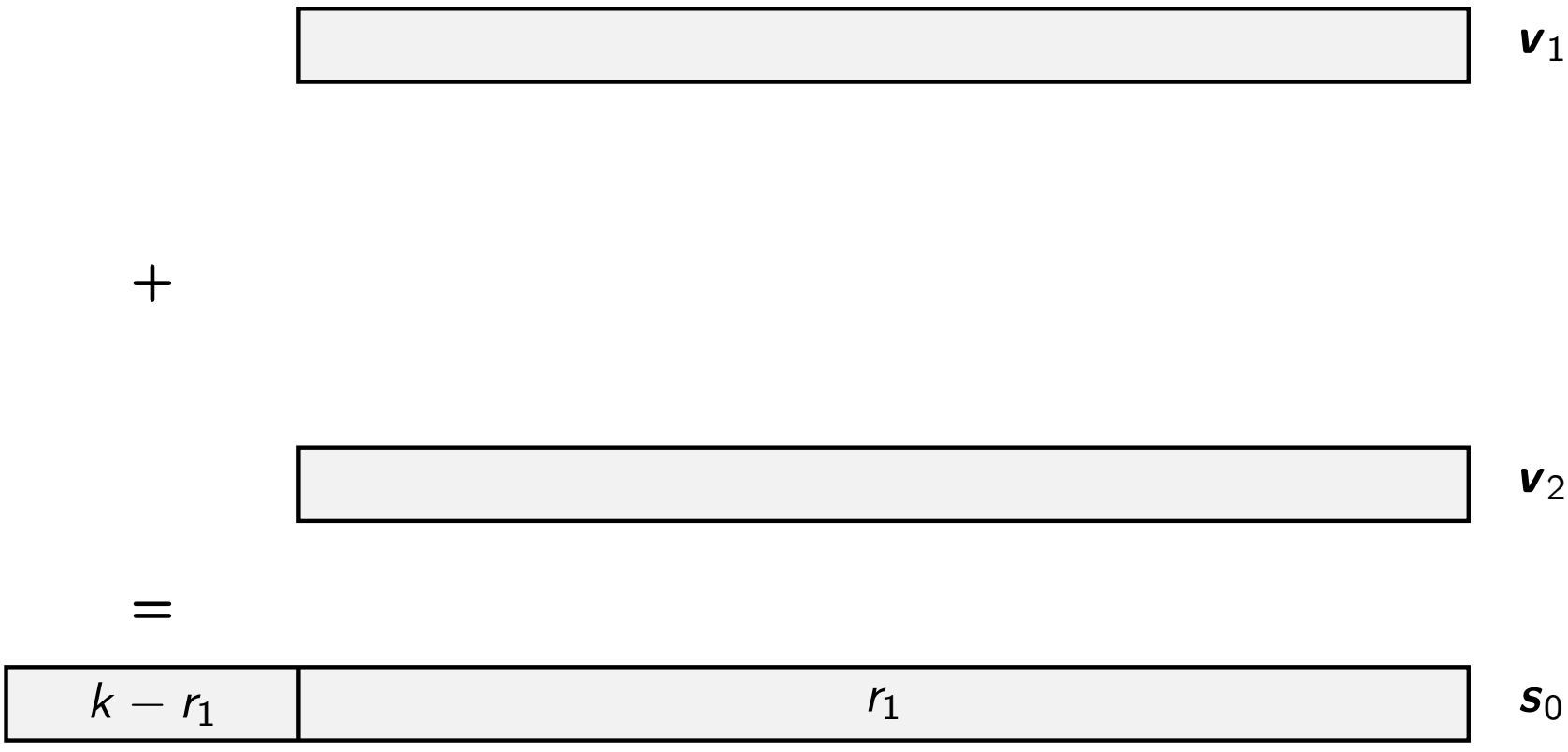
[11] Håvard Molland, John Erik Mathiassen, and Tor Helleseth.
Improved fast correlation attack using low rate codes.
In Kenneth G. Paterson, editor, *Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 67–81. Springer-Verlag, New York, NY, 2003.

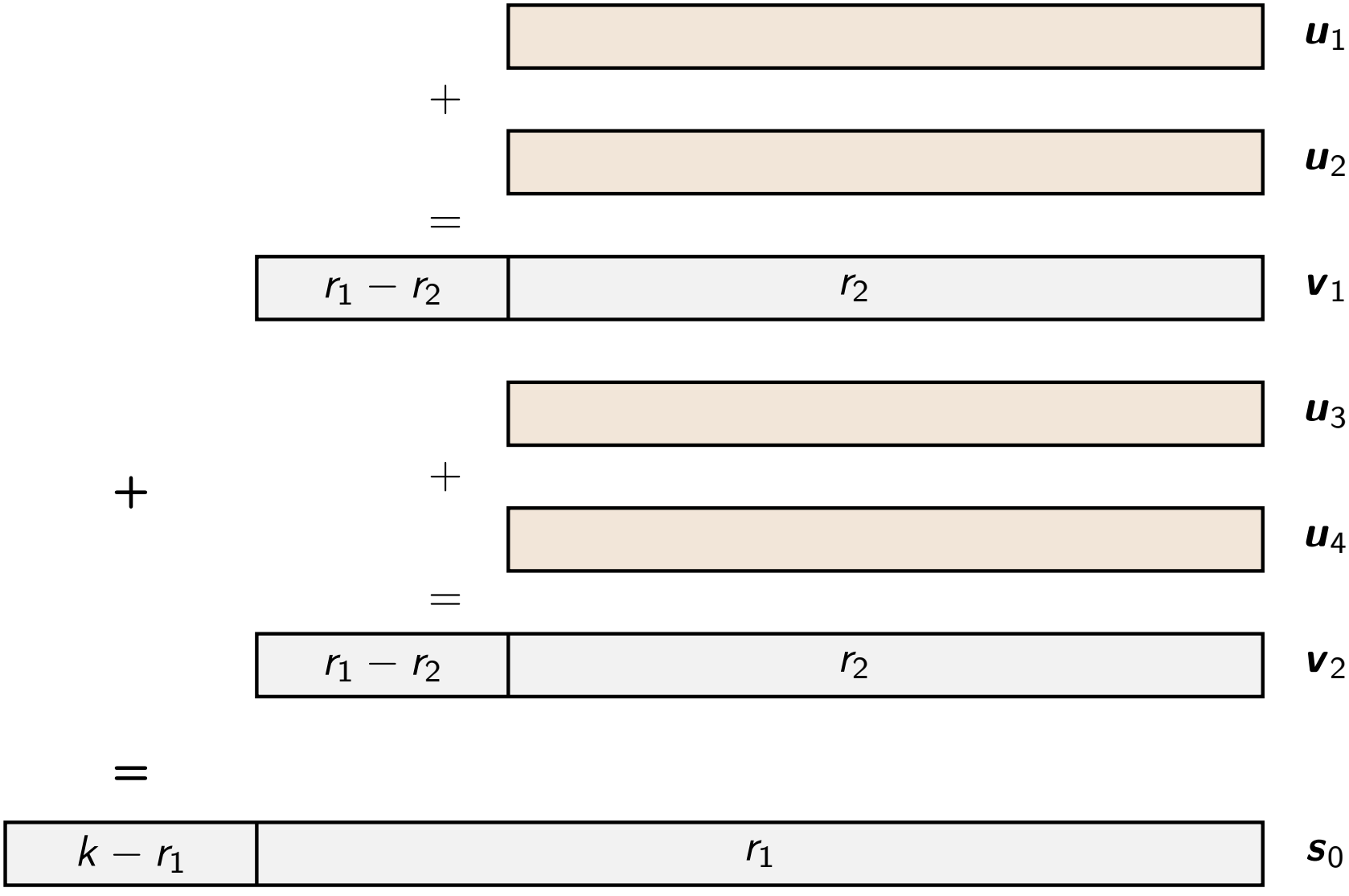
[12] Maneli Noorkami and Faramarz Fekri.
A fast correlation attack via unequal error correcting LDPC codes.
In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 54–46. Springer-Verlag, Berlin, 2004.

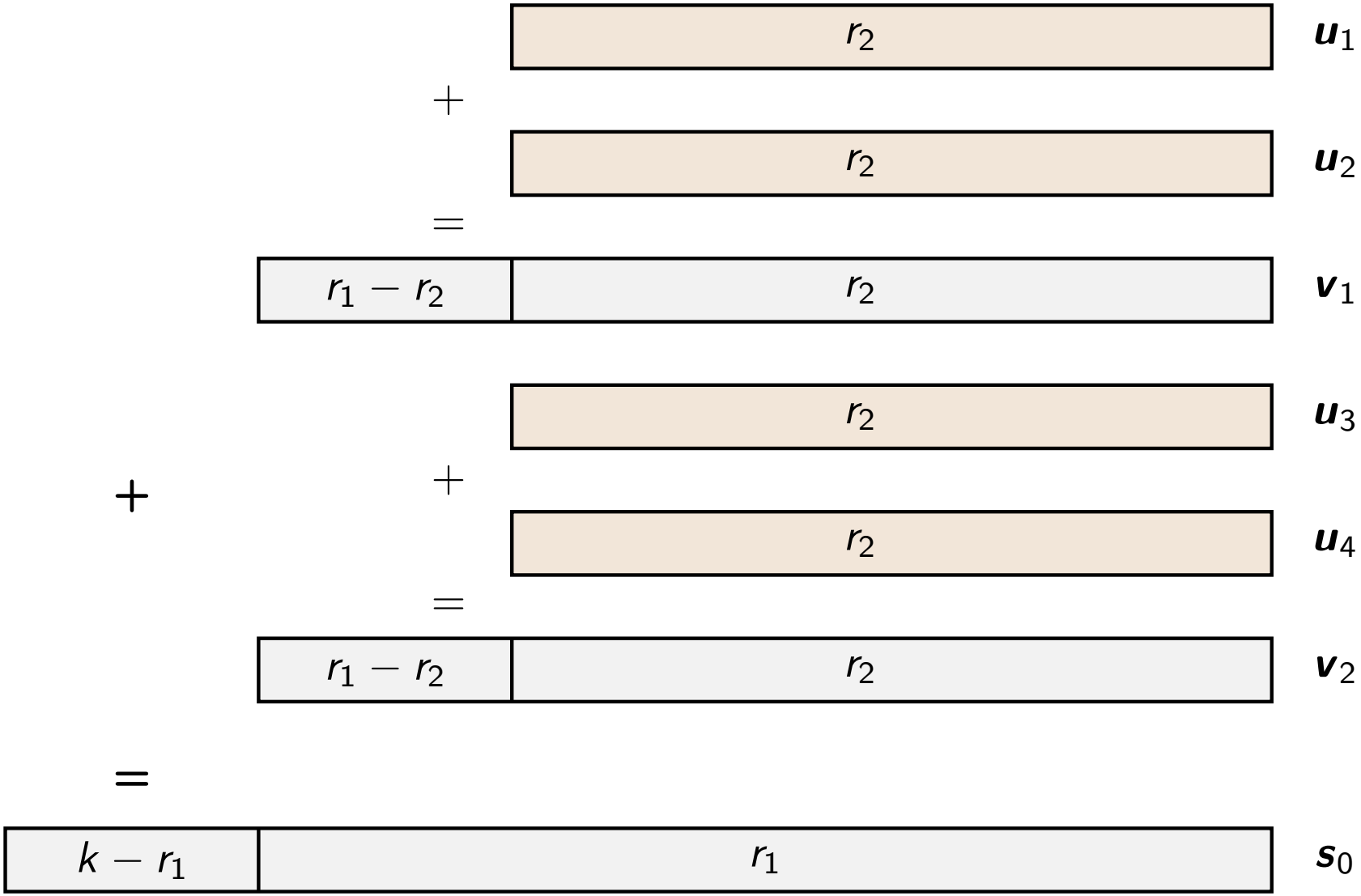
[13] Thomas Siegenthaler.
Decrypting a class of stream ciphers using ciphertext only.
IEEE Transactions on Computers, 34(1):81–85, 1985.

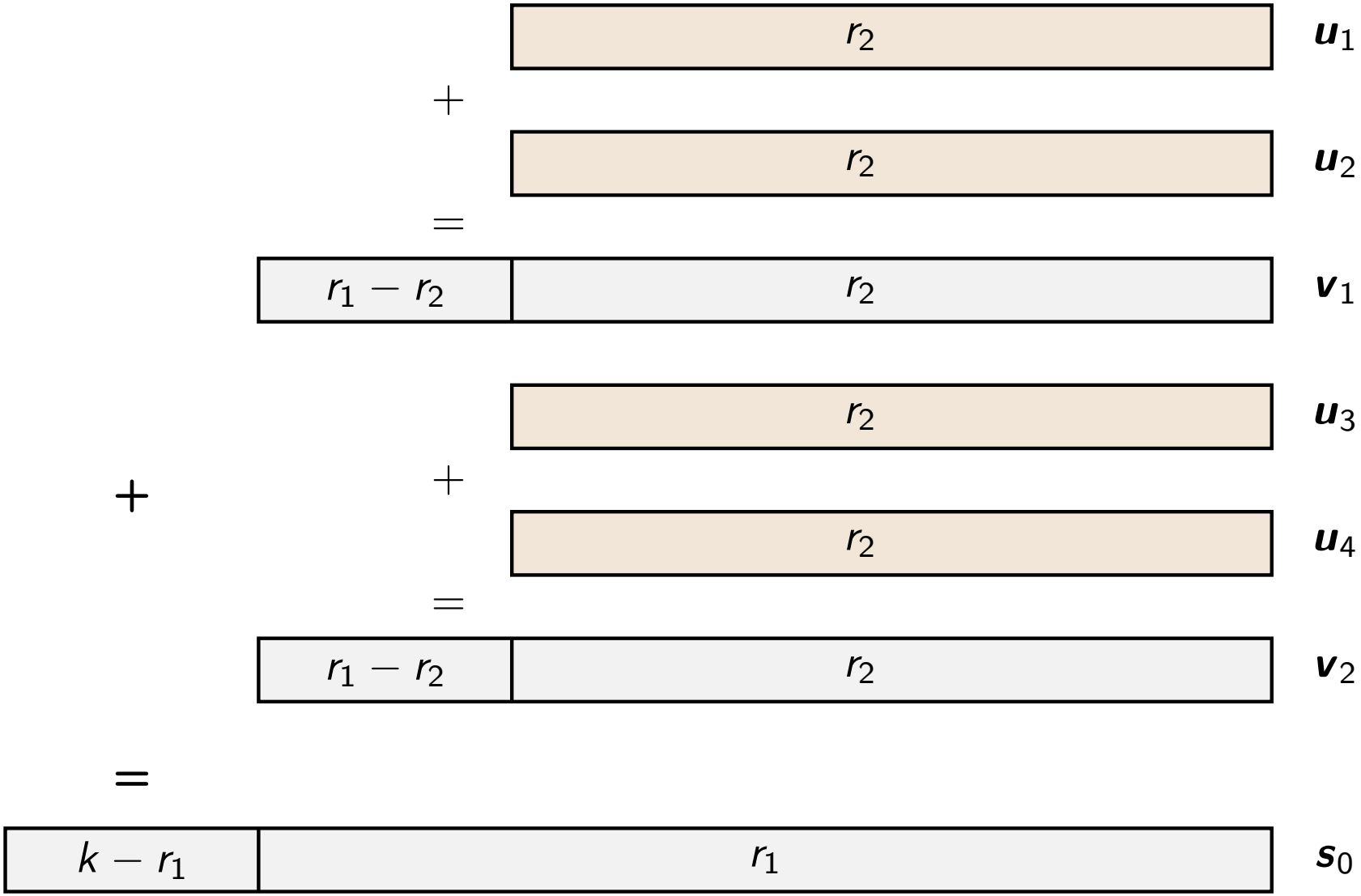
Thanks for your attention!


s_0


$\boldsymbol{v}_1$
+
$\boldsymbol{v}_2$
=
$k - r_1$
r_1
$\boldsymbol{s}_0$

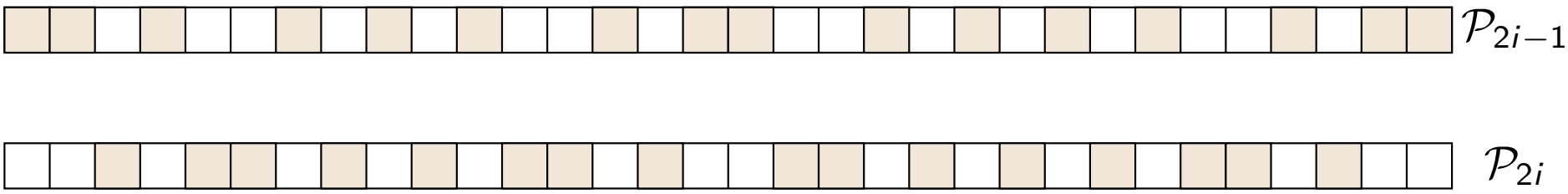

u_1
+
u_2
=
$r_1 - r_2$
r_2
v_1
u_3
+
+
u_4
=
$r_1 - r_2$
r_2
v_2
=
$k - r_1$
r_1
s_0


r_2
u_1
+
r_2
u_2
=
$r_1 - r_2$
r_2
v_1
r_2
u_3
+
+
r_2
u_4
=
$r_1 - r_2$
r_2
v_2
=
$k - r_1$
r_1
s_0

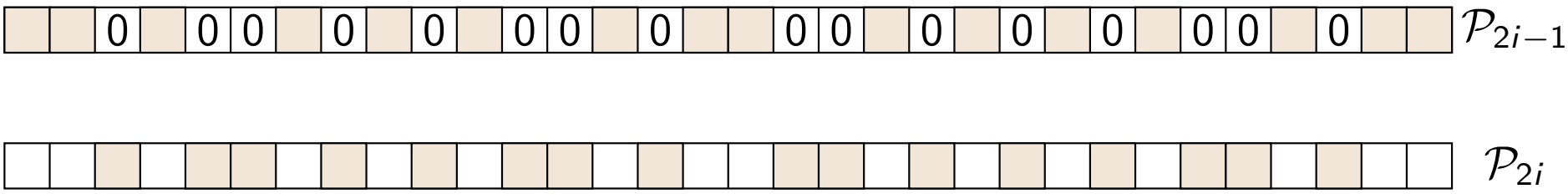


◂ Return

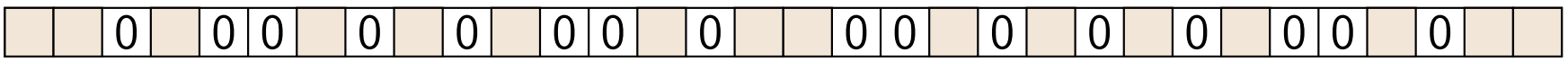
$k = 32$

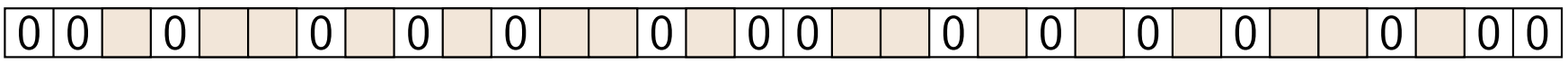


$k = 32$

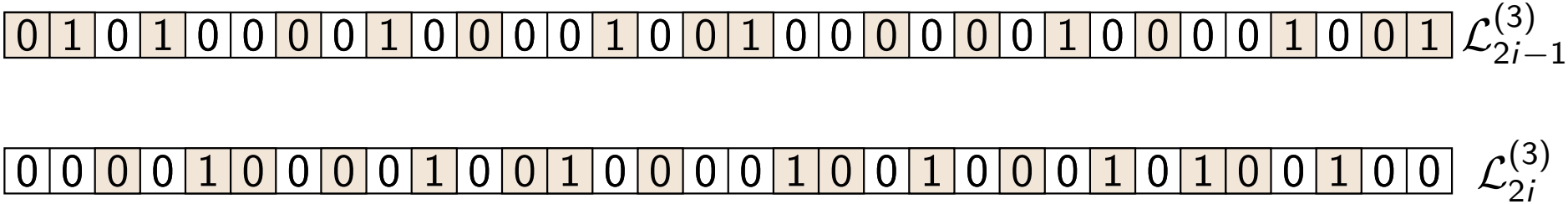

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\mathcal{P}_{2i-1}$
$\mathcal{P}_{2i}$

$k = 32$

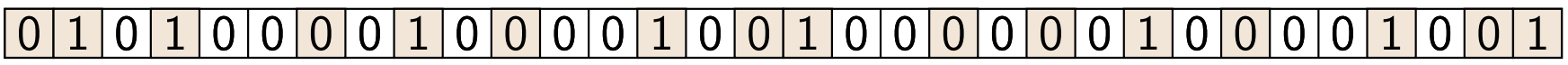




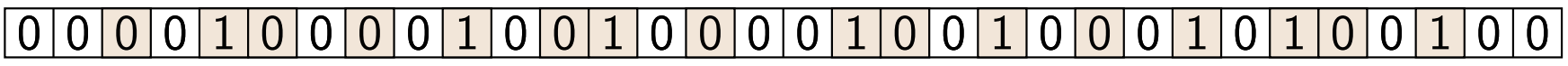
$k = 32$


0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1
$\mathcal{L}_{2i-1}^{(3)}$
0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0
$\mathcal{L}_{2i}^{(3)}$

$k = 32$



$\mathcal{L}_{2i-1}^{(3)}$



$\mathcal{L}_{2i}^{(3)}$

If $\boldsymbol{a}_1 \in \mathcal{L}_{2i-1}^{(3)}$ and $\boldsymbol{a}_2 \in \mathcal{L}_{2i}^{(3)}$, then $\mathrm{wt}(\boldsymbol{a}_1 + \boldsymbol{a}_2) = \mathrm{wt}(\boldsymbol{a}_1) + \mathrm{wt}(\boldsymbol{a}_2)$

◂ Return

$$\mathcal{L}_i^{(2)} = \Big\{ \boldsymbol{a}_1^{(i)} + \boldsymbol{a}_2^{(i)} \in \mathbb{F}_2^{2k} \mid \boldsymbol{a}_1^{(i)} \in \mathcal{L}_{2i-1}^{(3)}, \boldsymbol{a}_2^{(i)} \in \mathcal{L}_{2i}^{(3)}, \\ \mathrm{wt}\Big(\boldsymbol{a}_1^{(i)} + \boldsymbol{a}_2^{(i)}\Big) = p_2, \\ \Big((\boldsymbol{a}_1^{(i)} + \boldsymbol{a}_2^{(i)}) H_1^T\Big)[k - r_2 + 1 : k] = \boldsymbol{u}_i \Big\}$$

$$\mathcal{L}_j^{(1)} = \Big\{ \boldsymbol{b}_1^{(j)} + \boldsymbol{b}_2^{(j)} \in \mathbb{F}_2^{2k} \mid \boldsymbol{b}_1^{(j)} \in \mathcal{L}_{2j-1}^{(2)}, \boldsymbol{b}_2^{(j)} \in \mathcal{L}_{2j}^{(2)}, \\ \mathrm{wt}\Big(\boldsymbol{b}_1^{(j)} + \boldsymbol{b}_2^{(j)}\Big) = p_1, \\ \Big((\boldsymbol{b}_1^{(j)} + \boldsymbol{b}_2^{(j)}) H_1^T\Big)[k - r_1 + 1 : k] = \boldsymbol{v}_j \Big\}$$

$$\mathcal{L} = \Big\{ \boldsymbol{c}_1 + \boldsymbol{c}_2 \in \mathbb{F}_2^{2k} \mid \boldsymbol{c}_1 \in \mathcal{L}_1^{(1)}, \boldsymbol{c}_2 \in \mathcal{L}_2^{(1)}, \\ \mathrm{wt}(\boldsymbol{c}_1 + \boldsymbol{c}_2) = p, \ \big((\boldsymbol{c}_1 + \boldsymbol{c}_2) H_1^T\big) = \boldsymbol{s}_0 \Big\}$$

◂ Return