

# Stationary Syndrome Decoding for Improved PCGs

**Stan Peceny (Now at Stealth Software Technologies)**

Joint work with:

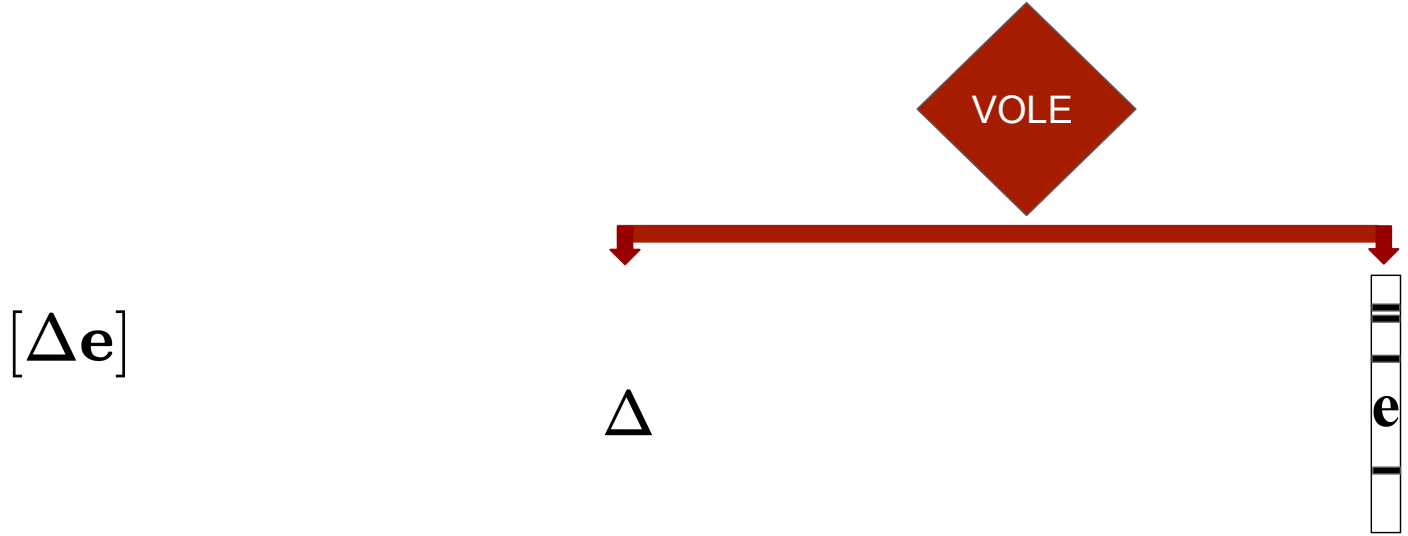
Vlad Kolesnikov, Srini Raghuraman, Peter Rindal



**Georgia Institute  
of Technology®**

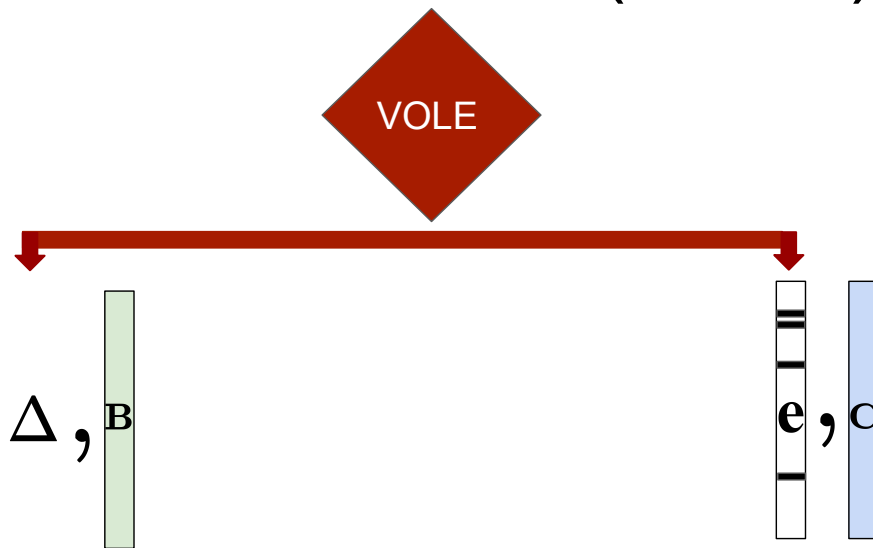


# Vector Oblivious Linear Evaluation (VOLE)

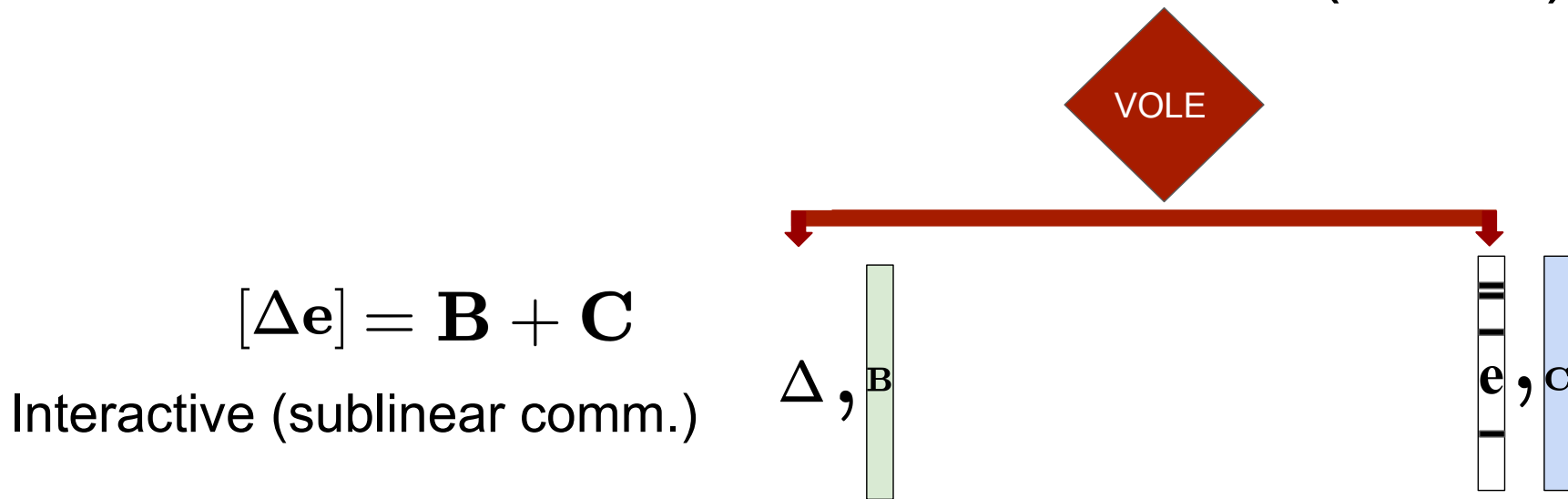


# Vector Oblivious Linear Evaluation (VOLE)

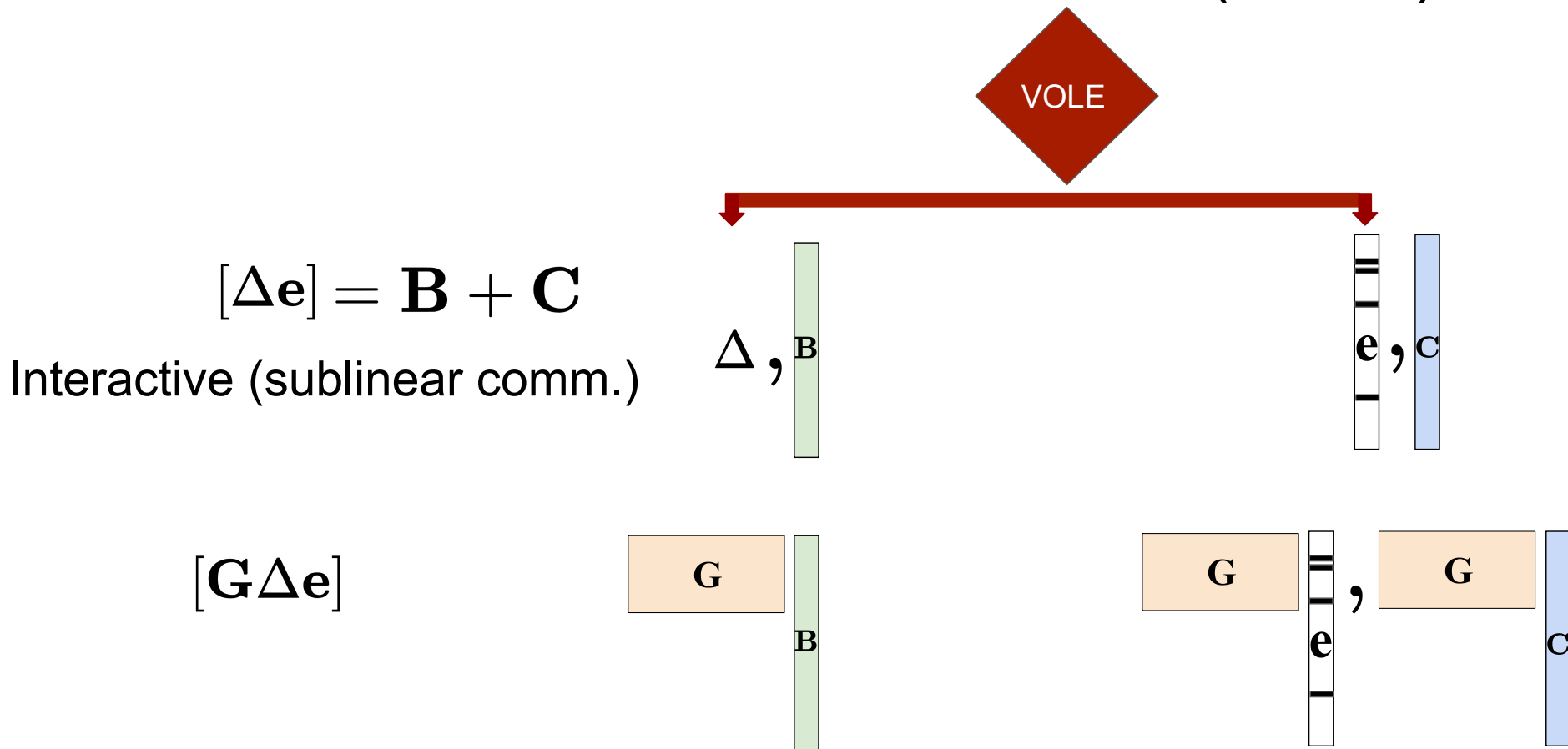
$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$



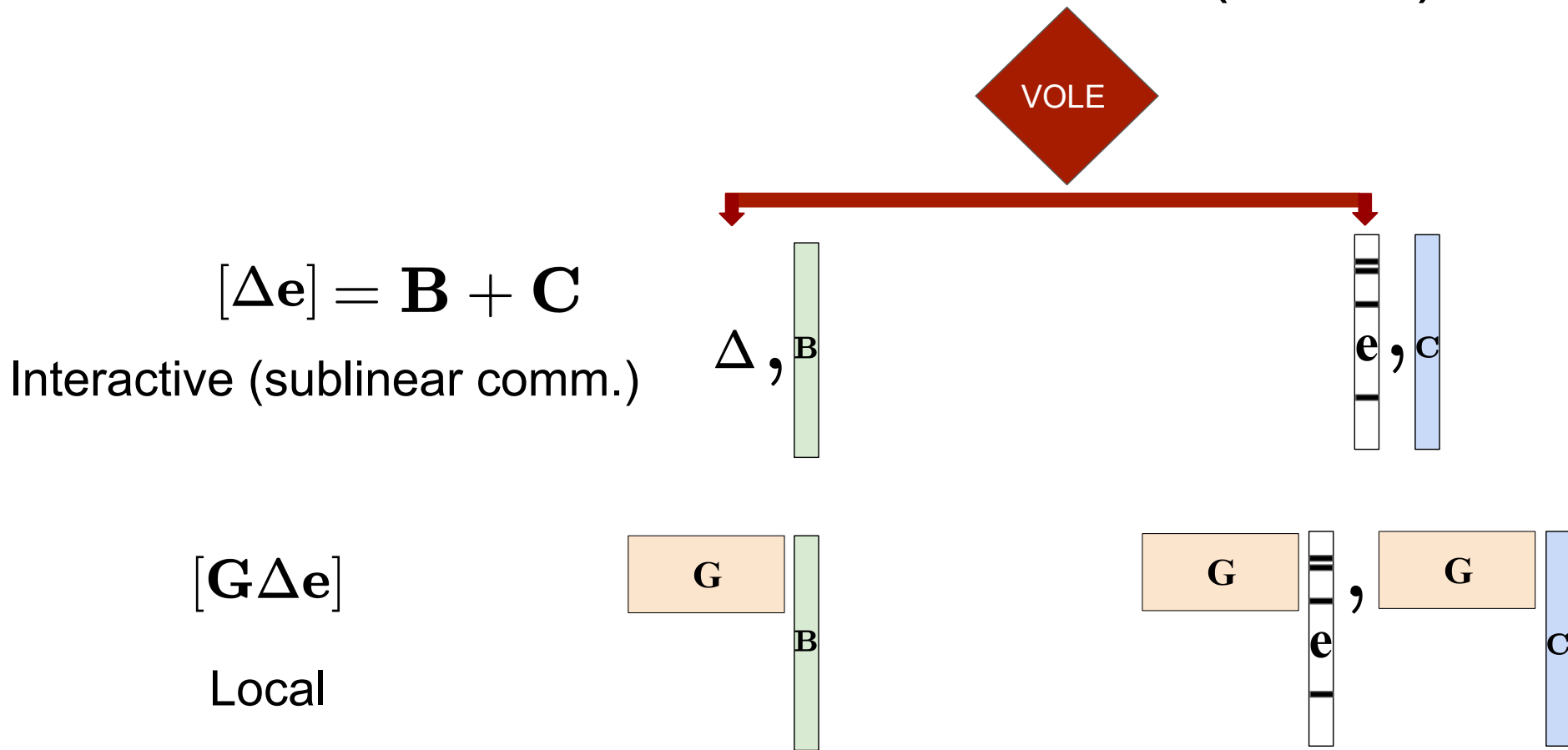
# Vector Oblivious Linear Evaluation (VOLE)



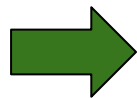
# Vector Oblivious Linear Evaluation (VOLE)



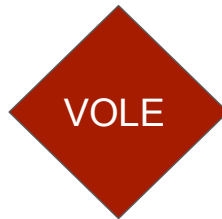
# Vector Oblivious Linear Evaluation (VOLE)



# Vector Oblivious Linear Evaluation (VOLE)

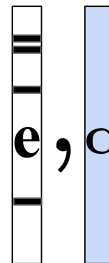
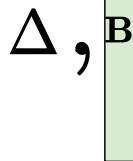


Similar for OT



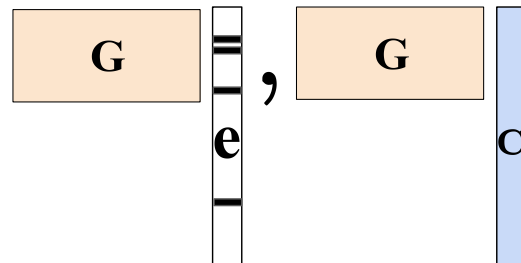
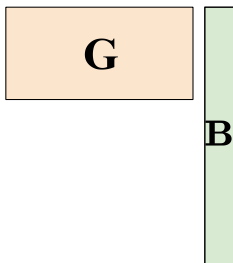
$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$

Interactive (sublinear comm.)

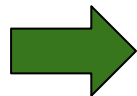


$$[\mathbf{G} \Delta \mathbf{e}]$$

Local

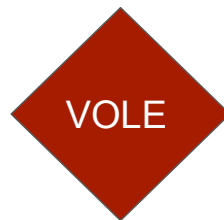


# Vector Oblivious Linear Evaluation (VOLE)



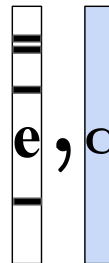
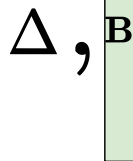
Similar for OT

Other correlations (Beaver triples)



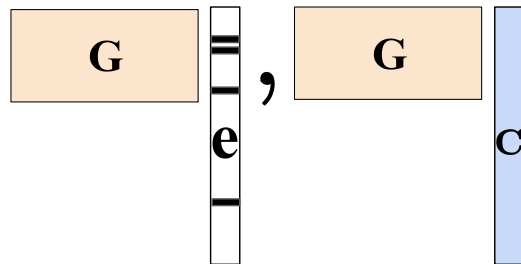
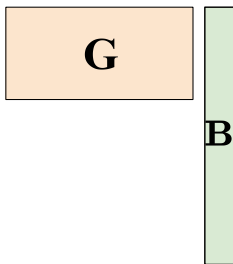
$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$

Interactive (sublinear comm.)



$$[\mathbf{G} \Delta \mathbf{e}]$$

Local



# Pseudorandom Correlation Generators (PCGs)

Sublinear communication, compelling computation

State of the art for generating correlated randomness

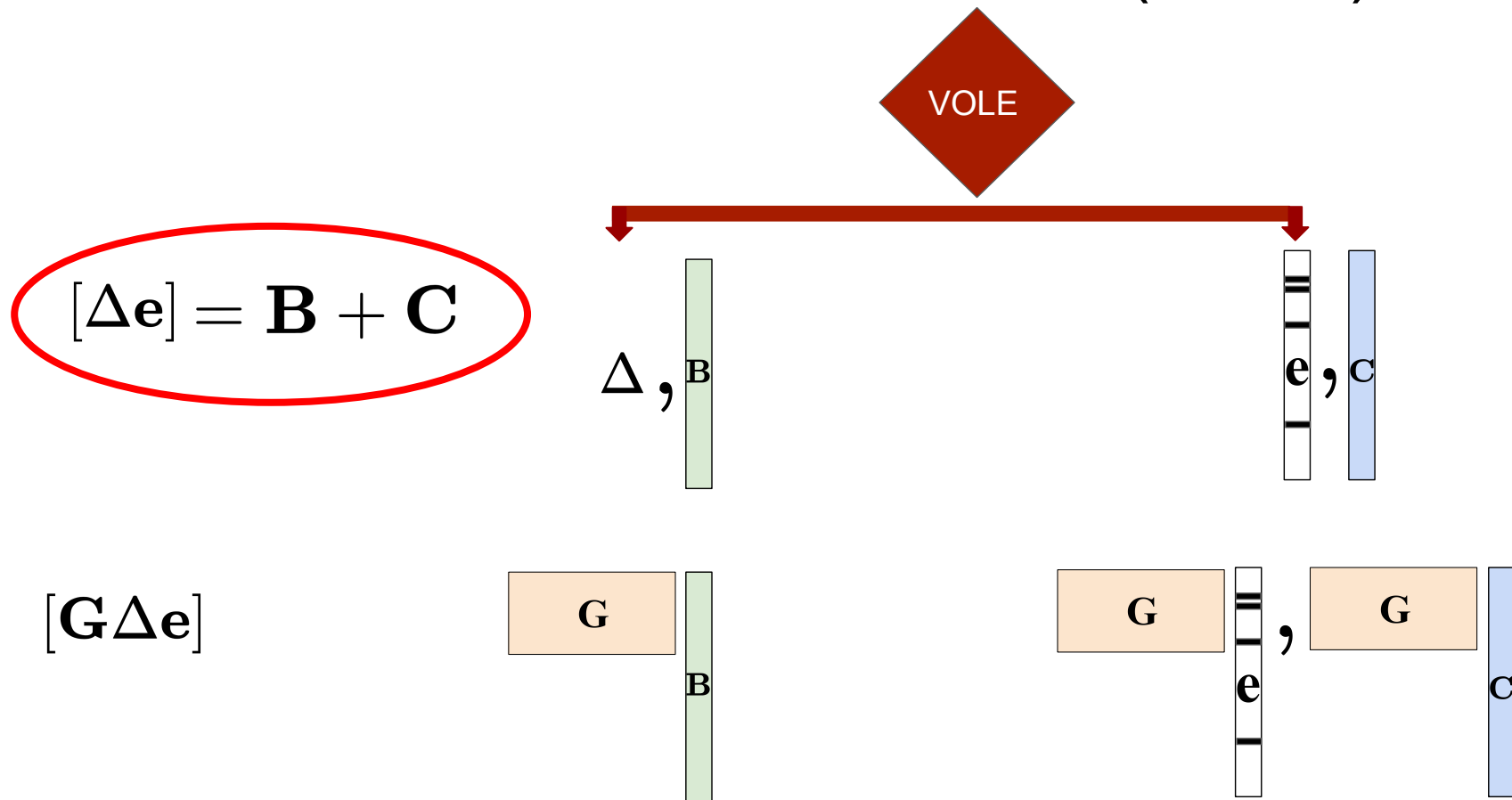
# Pseudorandom Correlation Generators (PCGs)

Sublinear communication, compelling computation

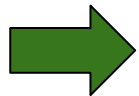
State of the art for generating correlated randomness

Correlated randomness is essential for MPC

# Vector Oblivious Linear Evaluation (VOLE)



# Vector Oblivious Linear Evaluation (VOLE)



Need many correlations for MPC

VOLE

$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$

$\Delta, \mathbf{B}$

$\mathbf{e}, \mathbf{C}$

$$[\mathbf{G}\Delta \mathbf{e}]$$

$\mathbf{G}$

$\mathbf{B}$

$\mathbf{G}$

$\mathbf{e}$

$\mathbf{G}$

$\mathbf{C}$

# Vector Oblivious Linear Evaluation (VOLE)

➔ Need many correlations for MPC

VOLE

$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$

Can we amortize this cost?

$\Delta, \mathbf{B}$

$\mathbf{e}, \mathbf{C}$

$$[\mathbf{G}\Delta \mathbf{e}]$$

$\mathbf{G}$

$\mathbf{B}$

$\mathbf{G}$

$\mathbf{e}$

$\mathbf{G}$

$\mathbf{C}$

LPN

Syndrome Decoding (SD)

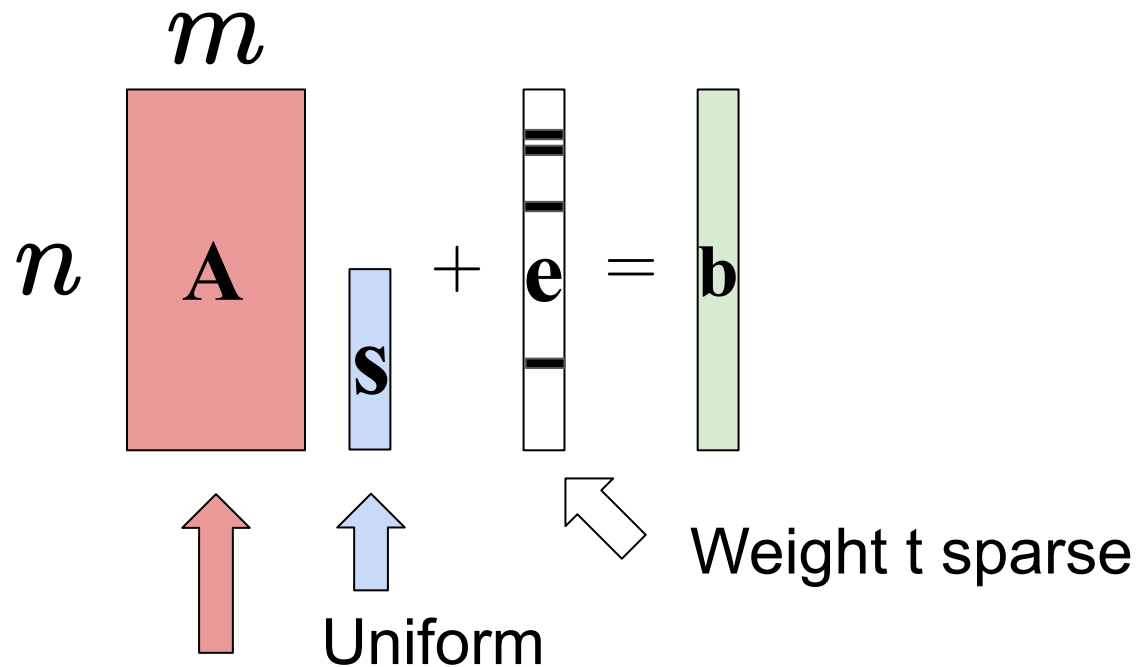
LPN

Syndrome Decoding (SD)

The diagram illustrates the Linear Programming (LPN) equation  $As = b$ . It features four vertical rectangular blocks representing matrices and vectors. The first block is a red rectangle labeled  $A$  with dimensions  $n$  (height) and  $m$  (width). The second block is a blue rectangle labeled  $s$ . The third block is a white rectangle labeled  $e$  with horizontal error bars, representing an error vector. The fourth block is a green rectangle labeled  $b$ . The equation is represented by the sequence of blocks:  $A$ ,  $s$ ,  $+$ ,  $e$ ,  $=$ ,  $b$ .

LPN

Syndrome Decoding (SD)

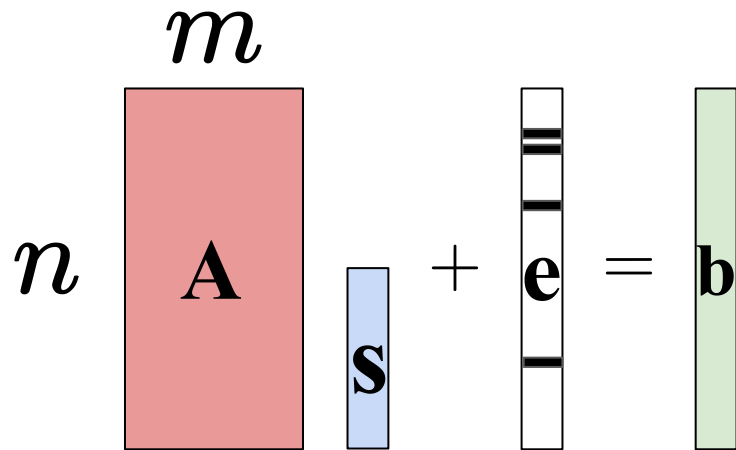


The diagram illustrates the LPN equation  $A \cdot s + e = b$ . Matrix  $A$  is a red rectangle with dimensions  $n$  (height) and  $m$  (width). Vector  $s$  is a blue rectangle. Vector  $e$  is a white rectangle with horizontal black bars, indicating sparsity. Vector  $b$  is a green rectangle. A red arrow points to  $A$  with the label "Transpose of a parity check matrix". A blue arrow points to  $s$  with the label "Uniform". A white arrow points to  $e$  with the label "Weight t sparse".

$$\begin{matrix} n \\ \uparrow \\ \text{Transpose of a parity check matrix} \end{matrix} \begin{matrix} m \\ \text{A} \end{matrix} + \begin{matrix} \text{Uniform} \\ \uparrow \\ s \end{matrix} + \begin{matrix} \text{Weight t sparse} \\ \swarrow \\ e \end{matrix} = b$$

LPN

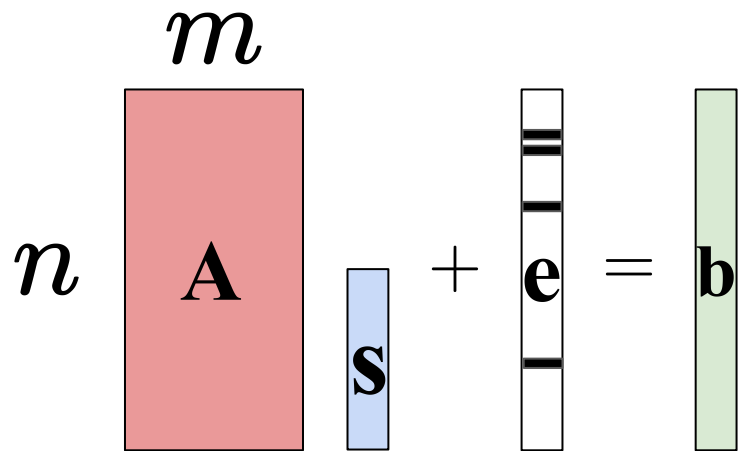
Syndrome Decoding (SD)



The diagram illustrates the Linear Polynomial Noise (LPN) equation. It features four vertical rectangles: a large red rectangle labeled  $\mathbf{A}$  with dimensions  $n$  (height) and  $m$  (width); a small blue rectangle labeled  $\mathbf{s}$ ; a tall white rectangle labeled  $\mathbf{e}$  with four horizontal black bars; and a tall green rectangle labeled  $\mathbf{b}$ . The equation is represented as  $\mathbf{A} \mathbf{s} + \mathbf{e} = \mathbf{b}$ , with a plus sign between  $\mathbf{s}$  and  $\mathbf{e}$ , and an equals sign between  $\mathbf{e}$  and  $\mathbf{b}$ .

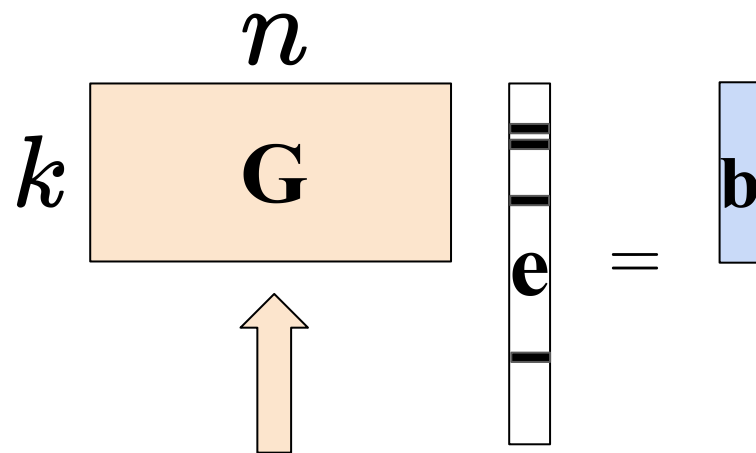
$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

## LPN



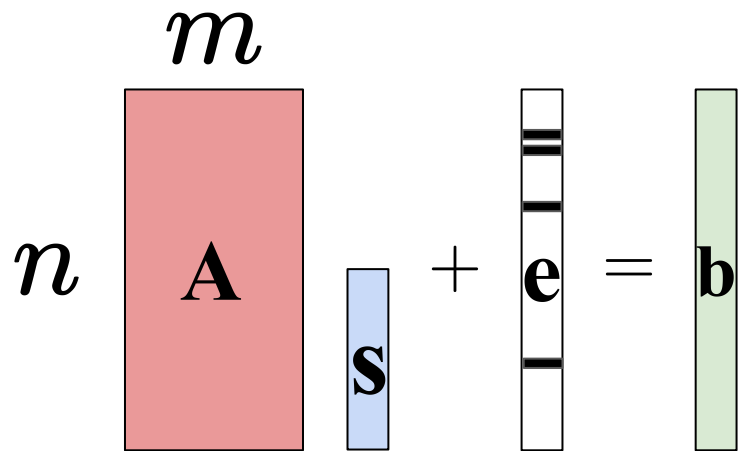
$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

## Syndrome Decoding (SD)



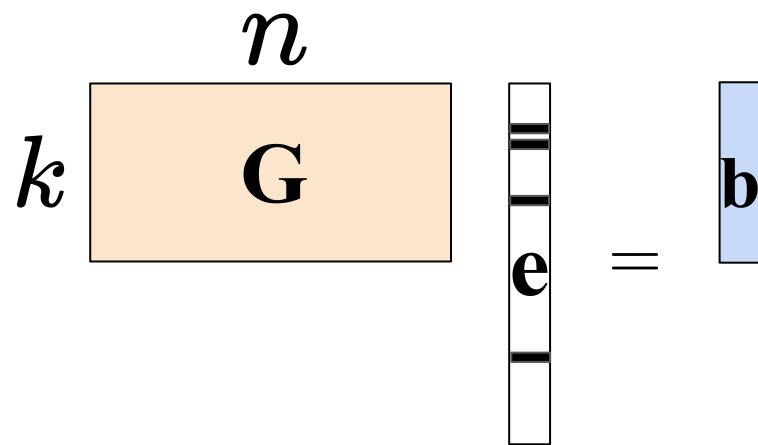
Generator

## LPN



$$(A, b) \approx (A, \$)$$

## Syndrome Decoding (SD)



$$(G, b) \approx (G, \$)$$

## LPN

$$\begin{matrix} m \\ \text{A} \\ n \end{matrix} + \begin{matrix} \text{s} \end{matrix} = \begin{matrix} \text{b} \end{matrix}$$

$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

LPN and SD are equivalent

## Syndrome Decoding (SD)

$$\begin{matrix} n \\ \text{G} \\ k \end{matrix} \begin{matrix} \text{e} \end{matrix} = \begin{matrix} \text{b} \end{matrix}$$

$$(\mathbf{G}, \mathbf{b}) \approx (\mathbf{G}, \$)$$

## LPN

$$\begin{matrix} m \\ n \end{matrix} \begin{matrix} \mathbf{A} \end{matrix} \begin{matrix} \mathbf{s} \end{matrix} + \begin{matrix} \mathbf{e} \end{matrix} = \begin{matrix} \mathbf{b} \end{matrix}$$

$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

LPN and SD are equivalent

## Syndrome Decoding (SD)

$$\begin{matrix} n \\ k \end{matrix} \begin{matrix} \mathbf{G} \end{matrix} \begin{matrix} \mathbf{e} \end{matrix} = \begin{matrix} \mathbf{b} \end{matrix}$$

$$(\mathbf{G}, \mathbf{b}) \approx (\mathbf{G}, \$)$$

Used for PCGs

# Syndrome Decoding (SD)

Known to be false for some choices of **G** and **e**

# Noise Distributions

**e**



Bernoulli - classic, sample  $e_i$  with  $\text{Ber}_{t/n}$

Exact - fixes Hamming weight to  $t$

# Noise Distributions

**e**



Bernoulli - classic, sample  $\mathbf{e}_i$  with  $\text{Ber}_{t/n}$

Exact - fixes Hamming weight to  $t$

**Regular** -  $t$  same-size blocks, each a random unit vector

# Noise Distributions

**e**



Bernoulli - classic, sample  $\mathbf{e}_i$  with  $\text{Ber}_{t/n}$

Exact - fixes Hamming weight to  $t$

**Regular** -  $t$  same-size blocks, each a random unit vector

**All of these improve for 1 instance**

# Noise Distributions

$\mathbf{e}$



Bernoulli - classic, sample  $\mathbf{e}_i$  with  $\text{Ber}_{t/n}$

Exact - fixes Hamming weight to  $t$

**Regular** -  $t$  same-size blocks, each a random unit vector

**All of these improve for 1 instance**

➡ We amortize the cost of  $[\Delta \mathbf{e}]$  across  $q$  SD instances

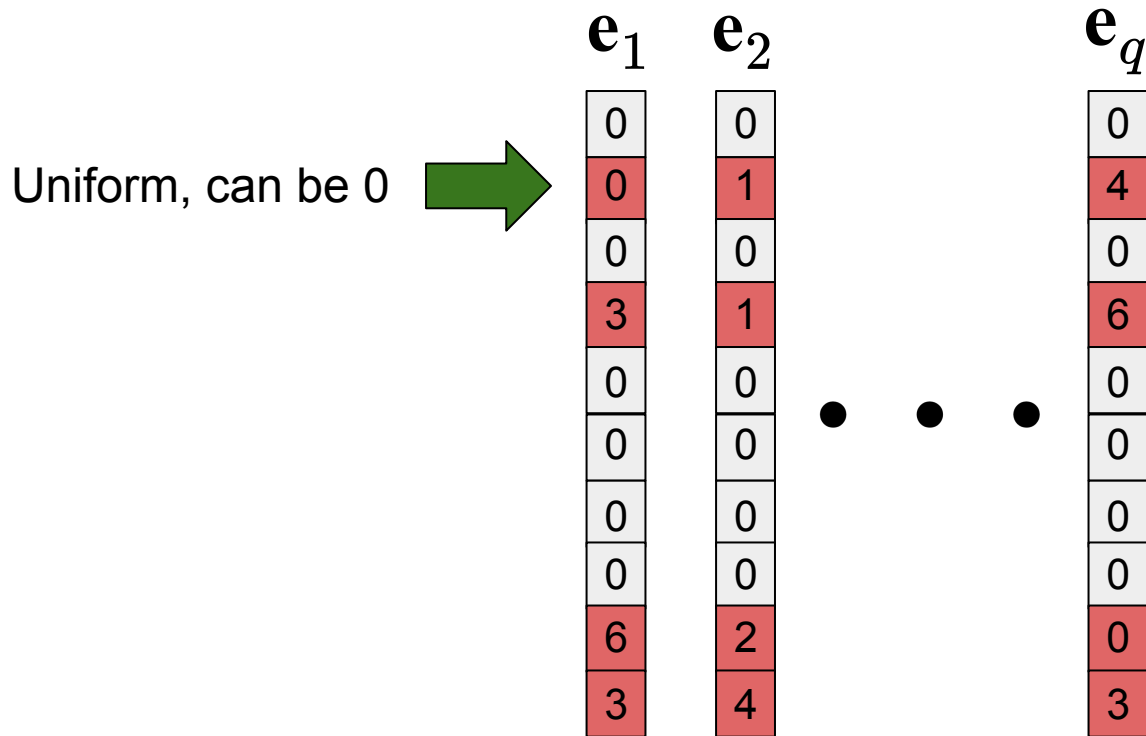
# Stationary Syndrome Decoding (SSD)

Noisy coordinates reused

| $\mathbf{e}_1$ | $\mathbf{e}_2$ |   | $\mathbf{e}_q$ |
|----------------|----------------|---|----------------|
| 0              | 0              |   | 0              |
| 0              | 1              |   | 4              |
| 0              | 0              |   | 0              |
| 3              | 1              |   | 6              |
| 0              | 0              | • | 0              |
| 0              | 0              | • | 0              |
| 0              | 0              | • | 0              |
| 0              | 0              |   | 0              |
| 6              | 2              |   | 0              |
| 3              | 4              |   | 3              |

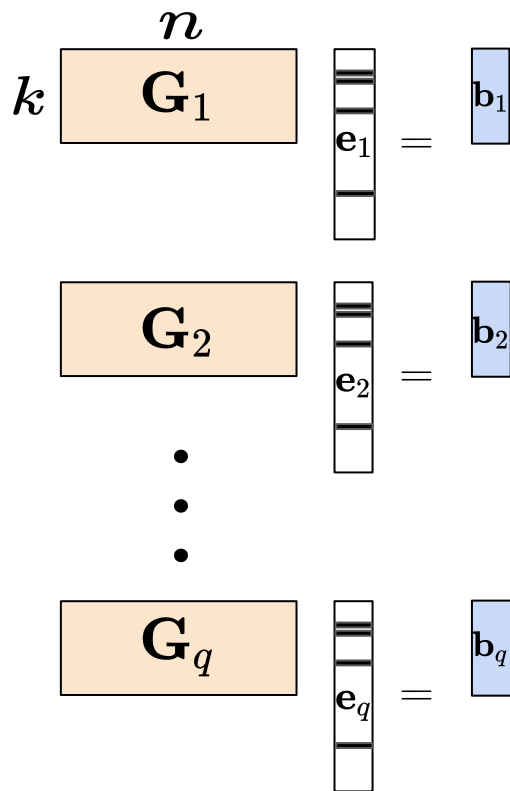
Noise in red in  $\mathbb{F}_7$

# Stationary Syndrome Decoding (SSD)



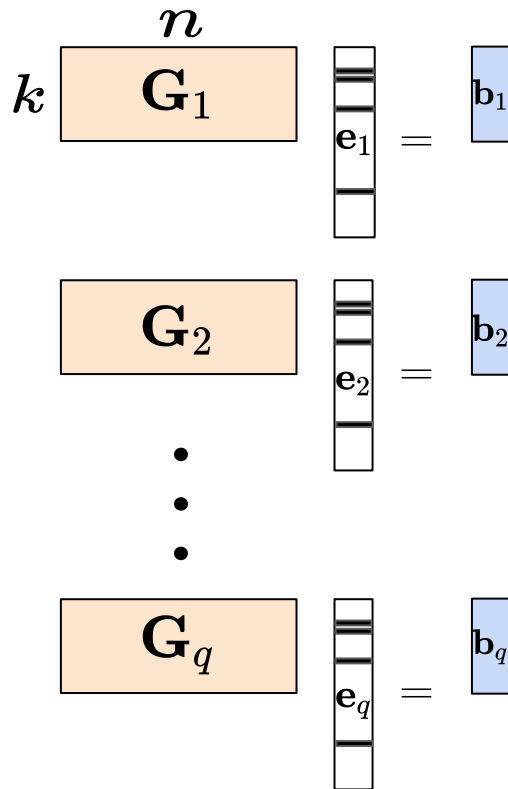
Noise in red in  $\mathbb{F}_7$

# Stationary Syndrome Decoding



$$(\mathbf{G}_i, \mathbf{b}_i) \approx (\mathbf{G}_i, \$)$$

# Stationary Syndrome Decoding



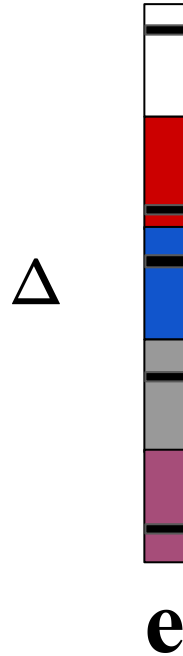
$$(\mathbf{G}_i, \mathbf{b}_i) \approx (\mathbf{G}_i, \$)$$

**We define SLPN similarly**

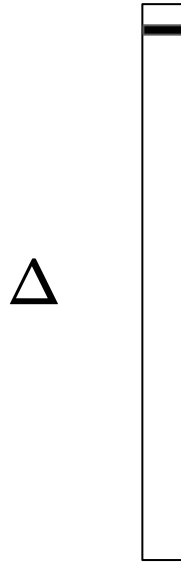
# Stationary Syndrome Decoding (SSD)

We cryptanalyze for  $\mathbf{G}_i$   
with high minimum distance and regular  $\mathbf{e}_i$

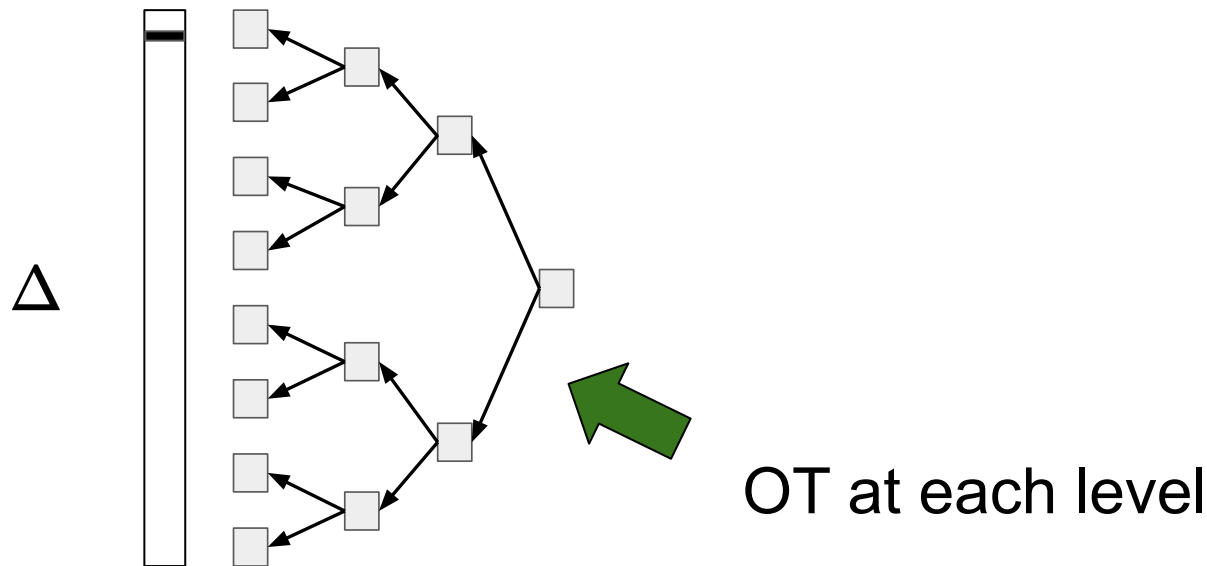
# How Does SSD Help with PCGs?



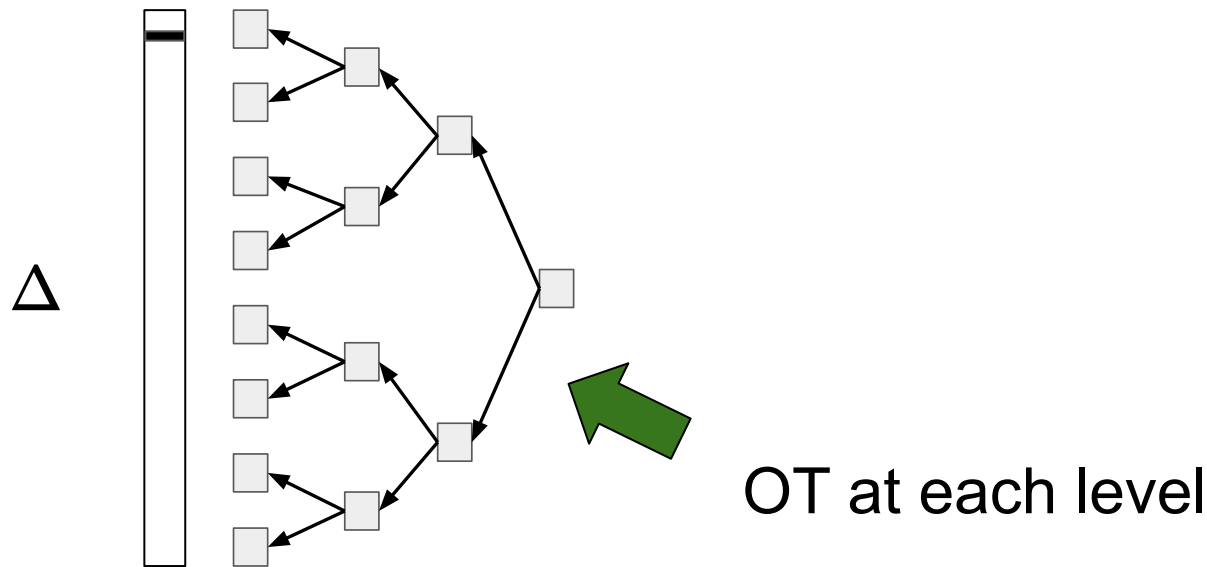
# How Does SSD Help with PCGs?



# How Does SSD Help with PCGs?

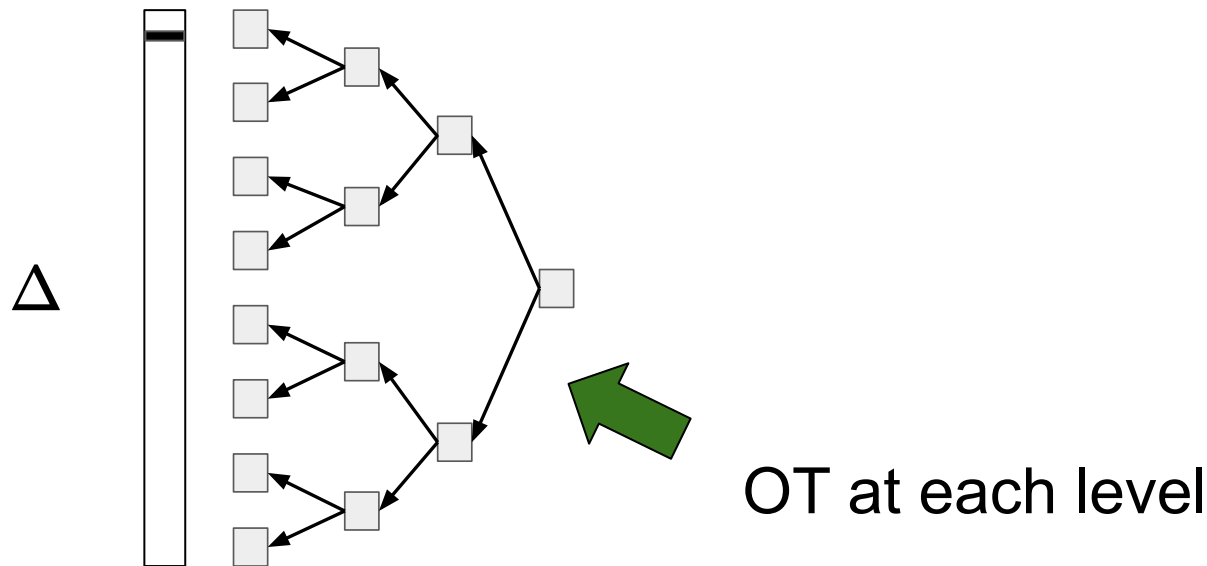


# How Does SSD Help with PCGs?



SSD allows for reusing OTs across all  $q$  noise vectors

# How Does SSD Help with PCGs?



SSD allows for reusing OTs across all  $q$  noise vectors

Better cache and memory utilization

# Presentation Outline

SSD's Resilience to Linear Attacks

Other Linear Attacks

SSD's Resilience to Algebraic Attacks

Experimental Evaluation

# Linear Attacks

Gaussian Eliminations [BKW00, Lyu05, LF06, EKM17]

Information Set Decoding [Pra62, Ste88, FS09, BLP11, MMT11, BJMM12, MO15, EKM17, BM18]

Cover Sets [ZW16, BV16, BTV16, GJL20]

Statistical Decoding Attacks [AJ01, FKI06, Ove06, DAT17]

Generalized Birthday Attacks [Wag02, Kir11]

Linearization Attacks [BM97, Saa07]

Low Weight Code [Zic17]

...

# Linear Attacks

Gaussian Eliminations [BKW00, Lyu05, LF06, EKM17]

Information Set Decoding [Pra62, Ste88, FS09, BLP11, MMT11, BJMM12, MO15, EKM17, BM18]

Cover Sets [ZW16, BV16, BTV16, GJL20]

Statistical Decoding Attacks [AJ01, FKI06, Ove06, DAT17]

Generalized Birthday Attacks [Wag02, Kir11]

Linearization Attacks [BM97, Saa07]

Low Weight Code [Zic17]

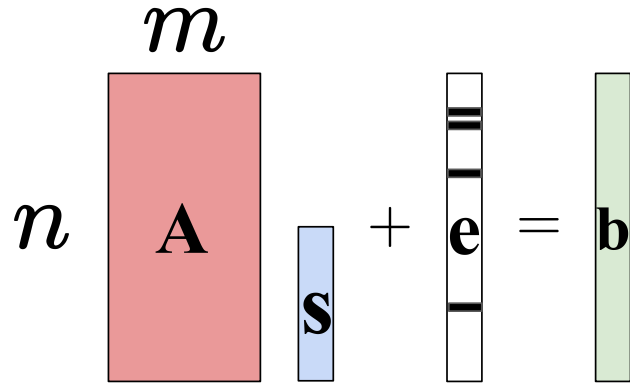
...



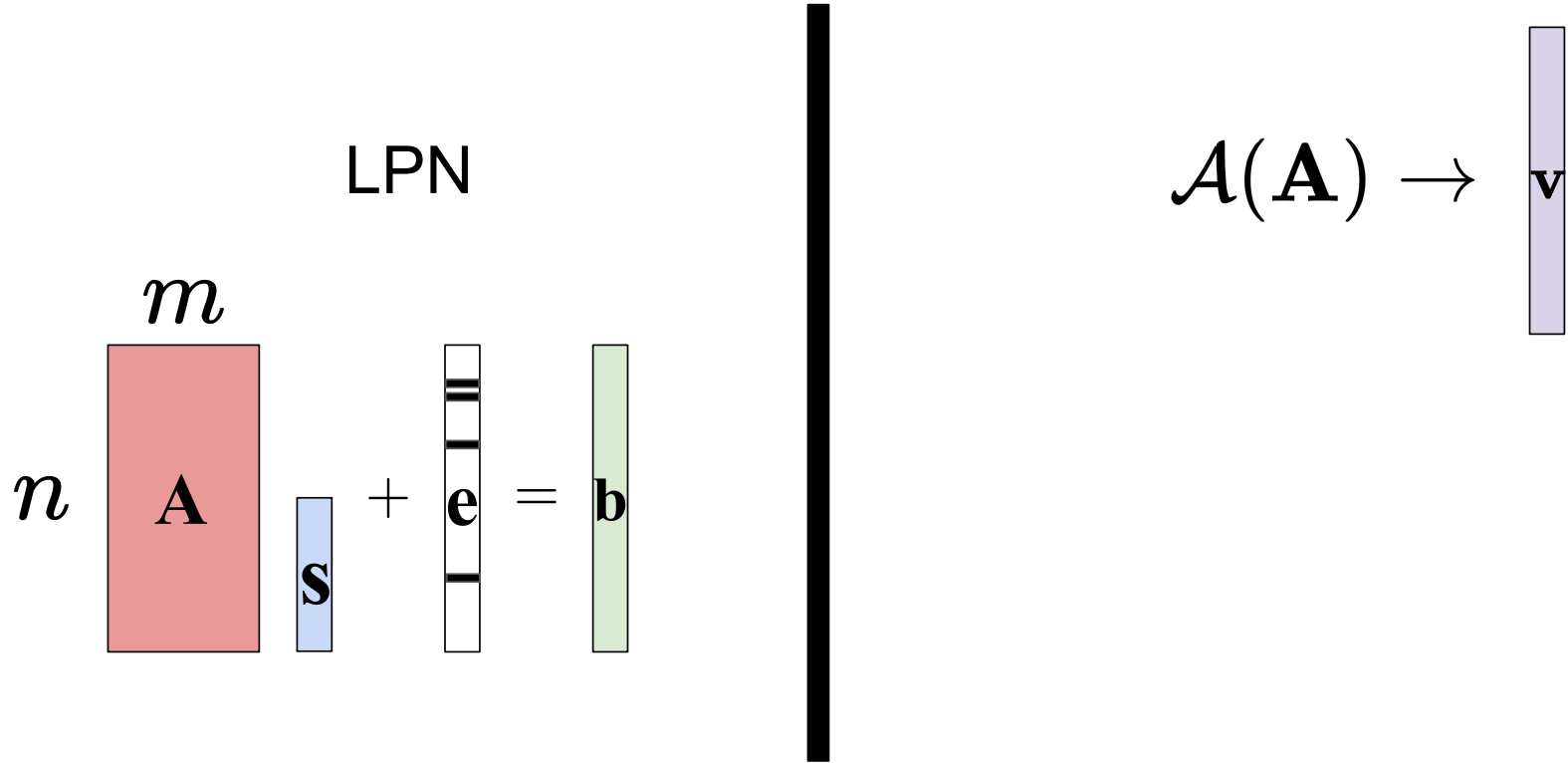
**Tedious to go through each attack**

# Linear Test Framework

LPN

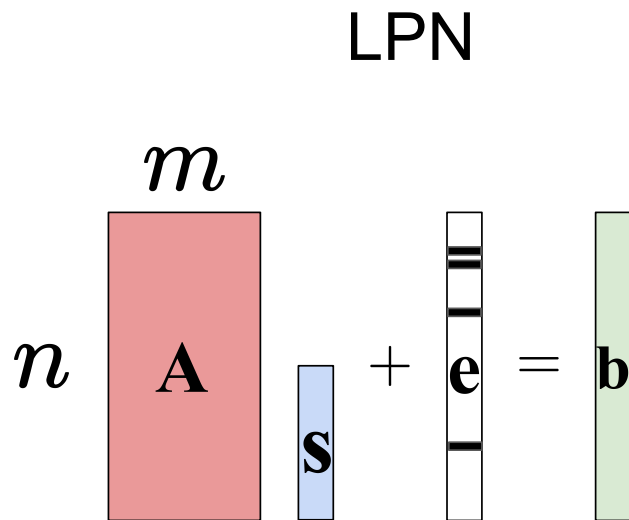


# Linear Test Framework



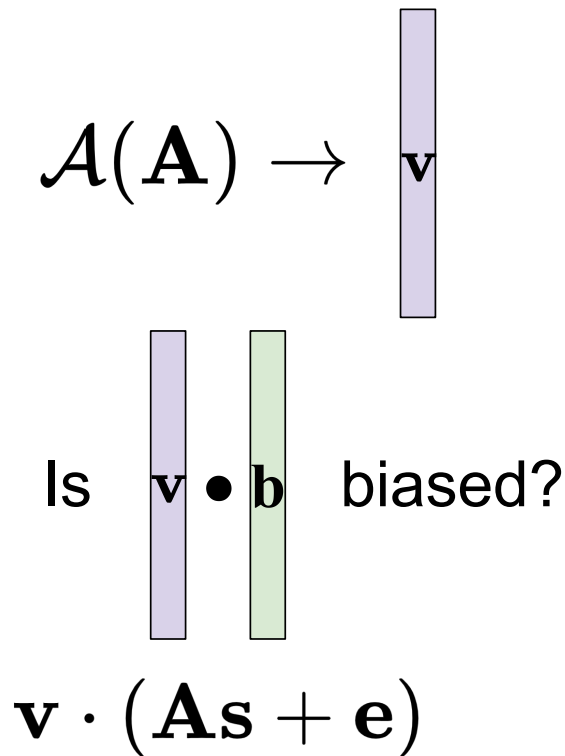
# Linear Test Framework

LPN

$$\begin{matrix} n & m \\ \mathbf{A} \end{matrix} \begin{matrix} \mathbf{s} \end{matrix} + \begin{matrix} \mathbf{e} \end{matrix} = \begin{matrix} \mathbf{b} \end{matrix}$$


$$\mathcal{A}(\mathbf{A}) \rightarrow \begin{matrix} \mathbf{v} \end{matrix}$$

Is  $\mathbf{v} \bullet \mathbf{b}$  biased?

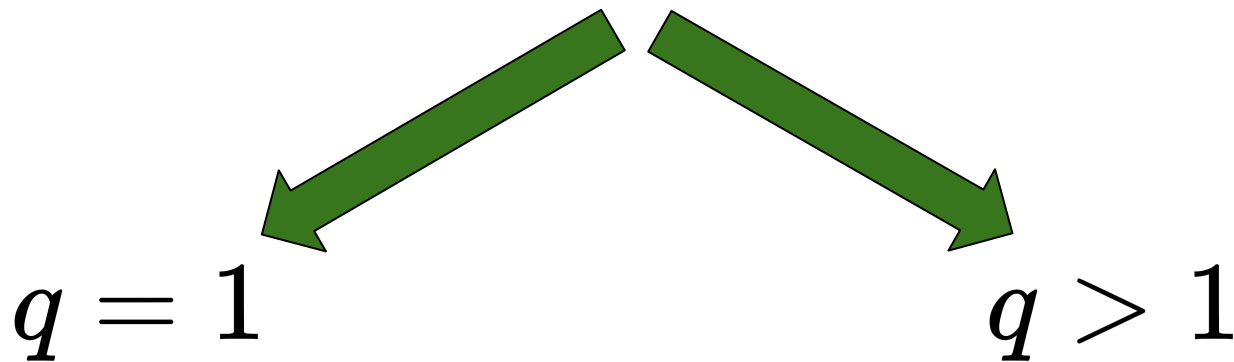
$$\mathbf{v} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e})$$


# High-Level Proof Template

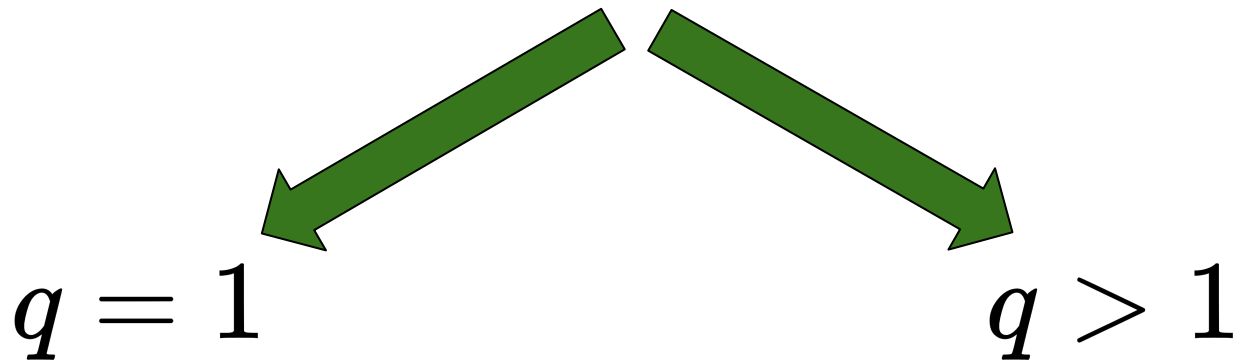
For SLPN with **regular** noise

Given equivalence of SLPN and SSD, security for SSD is straightforward

# High-Level Proof Template



# High-Level Proof Template



Differs from plain LPN  
with regular noise

# High-Level Proof Template

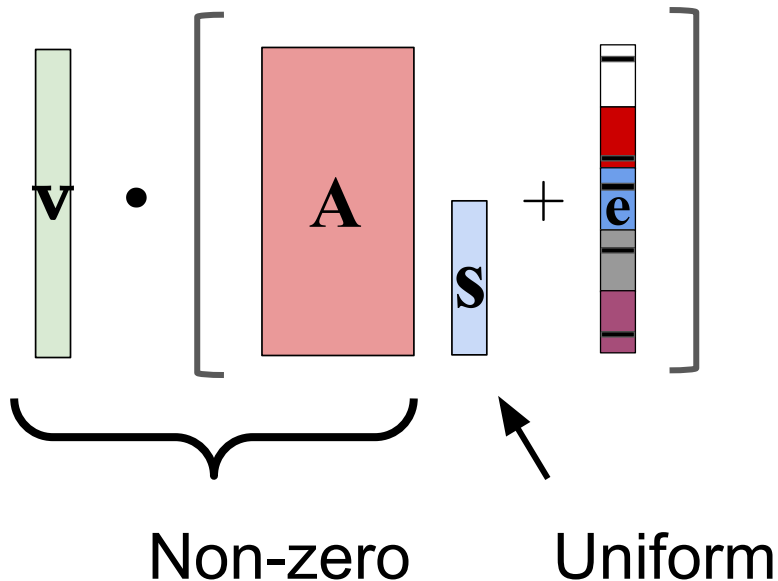
$$q = 1$$

non-codeword  $\mathbf{v}$

# High-Level Proof Template

$$q = 1$$

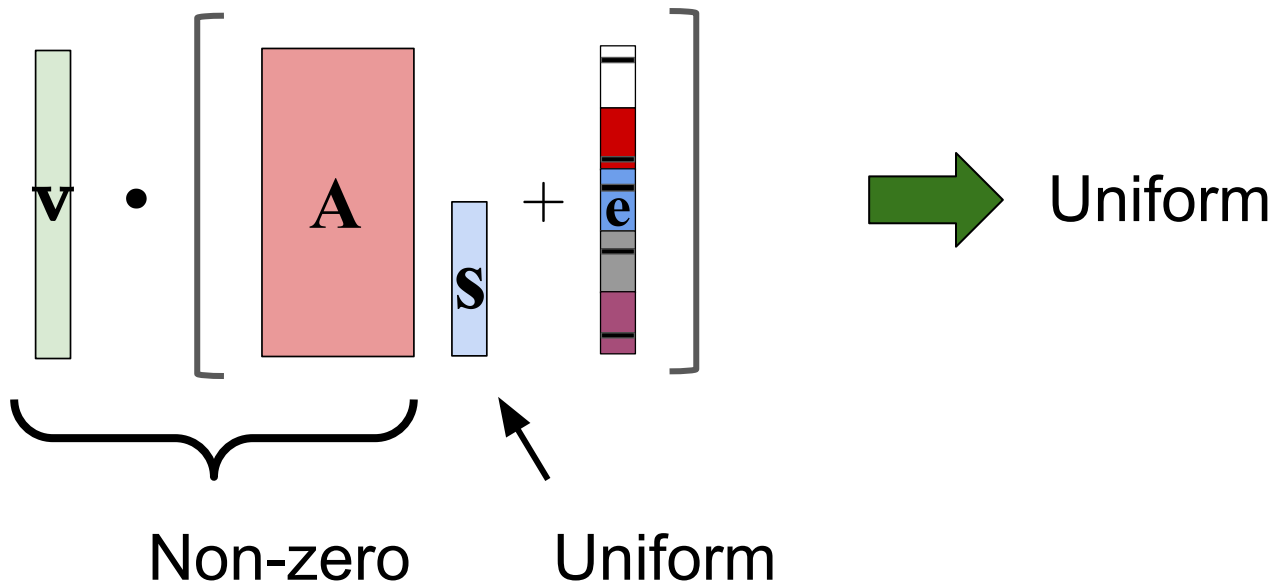
non-codeword  $\mathbf{v}$



# High-Level Proof Template

$$q = 1$$

non-codeword  $\mathbf{v}$



# High-Level Proof Template

$$q = 1$$

codeword  $\mathbf{v}$

# High-Level Proof Template

$$q = 1$$

codeword  $\mathbf{v}$

The diagram illustrates the equation  $\mathbf{v} \cdot [\mathbf{A} \mathbf{s} + \mathbf{e}]$ . On the left, a light green vertical rectangle is labeled  $\mathbf{v}$ . To its right is a dot  $\cdot$ . This is followed by a large square bracket containing three terms: a red rectangle labeled  $\mathbf{A}$ , a light blue rectangle labeled  $\mathbf{s}$ , and a plus sign  $+$  followed by a vertical stack of colored rectangles (white, red, blue, grey, purple) labeled  $\mathbf{e}$ . A curly brace is positioned below the  $\mathbf{v}$  rectangle and the opening of the square bracket.

Zero, randomness by  $\mathbf{s}$  vanishes

# High-Level Proof Template

$$q = 1$$

codeword  $\mathbf{v}$

The diagram illustrates the equation  $\mathbf{v} \cdot [\mathbf{A}\mathbf{s} + \mathbf{e}] = \mathbf{v} \cdot \mathbf{e}$  using colored blocks to represent vectors and matrices. On the left, a green vertical bar labeled  $\mathbf{v}$  is followed by a dot and a large square bracket. Inside the bracket is a red rectangle labeled  $\mathbf{A}$ , followed by a blue vertical bar labeled  $\mathbf{s}$ , a plus sign, and a vertical bar representing a vector  $\mathbf{e}$  with segments of white, red, blue, grey, and purple. The entire bracketed expression is followed by an equals sign. On the right, the same green vertical bar  $\mathbf{v}$  is followed by a dot and a vertical bar representing the vector  $\mathbf{e}$  with the same color segments. A curly brace is positioned below the left side of the equation, spanning from the green bar  $\mathbf{v}$  to the start of the square bracket.

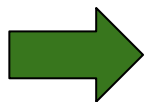
Zero, randomness by  $\mathbf{s}$  vanishes

# High-Level Proof Template

$$q = 1$$

codeword  $\mathbf{v}$

$$\mathbf{v} \cdot \left[ \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \right] = \mathbf{v} \cdot \mathbf{e}$$



Need to show  $\mathbf{v} \cdot \mathbf{e}$  has negligible bias

# High-Level Proof Template

$$q = 1$$

codeword  $\mathbf{v}$

$$\mathbf{v} \cdot \left[ \mathbf{A} + \mathbf{s} + \mathbf{e} \right] = \mathbf{v} \cdot \mathbf{e}$$
$$\leq \left( 1 - \frac{d}{n} \right)^t$$

# High-Level Proof Template

$$q = 1$$

codeword  $\mathbf{v}$

$$\mathbf{v} \cdot \left[ \mathbf{A} \mathbf{s} + \mathbf{e} \right] = \mathbf{v} \cdot \mathbf{e}$$

Regular LPN with  
 $\mathbb{F}_2$  noise

$$\leq \left( 1 - \frac{2d}{n} \right)^t$$

# High-Level Proof Template

Consider canonical representation for  $q > 1$ :

The diagram illustrates a matrix equation in a canonical representation. On the left is a large square matrix with a block diagonal structure. The diagonal blocks are labeled  $A_1, A_2, \dots, A_q$  and are highlighted in red. A bracket above the first block  $A_1$  is labeled  $m$ , indicating its width. A bracket to the left of the first block is labeled  $n$ , indicating its height. To the right of the matrix is a vertical stack of blue blocks labeled  $s_1, s_2, \dots, s_q$ . This is followed by a plus sign  $+$ . To the right of the plus sign is a vertical stack of white blocks labeled  $e_1, e_2, \dots, e_q$ . This is followed by an equals sign  $=$ . To the right of the equals sign is a vertical stack of green blocks labeled  $b_1, b_2, \dots, b_q$ .

$$\begin{bmatrix} \overbrace{A_1}^m & & \\ & A_2 & \\ & & \ddots \\ & & & A_q \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_q \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_q \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{bmatrix}$$

# High-Level Proof Template

$$q > 1$$

$\mathbf{v}$  is not a concatenation of  $q$  codewords

# High-Level Proof Template

$$q > 1$$

$\mathbf{v}$  is not a concatenation of  $q$  codewords

$\mathbf{v} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e})$  is uniform because  $\mathbf{s}$  is not mapped to 0

# High-Level Proof Template

$$q > 1$$

$\mathbf{v}$  is a concatenation of  $q$  codewords

# High-Level Proof Template

$$q > 1$$

$\mathbf{v}$  is a concatenation of  $q$  codewords

$$\mathbf{v} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e}) = \mathbf{v} \cdot \mathbf{e}$$

# High-Level Proof Template

$$q > 1$$

$\mathbf{v}$  is a concatenation of  $q$  codewords

$$\mathbf{v} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e}) = \mathbf{v} \cdot \mathbf{e}$$

$$\leq \left(1 - \frac{d}{n}\right)^t$$

# Other Linear Attacks

Explored new attacks that could be considered linear but do not fit into the linear test framework

# Algebraic Attacks

Solve for  $\mathbf{e}_1, \dots, \mathbf{e}_q$  in a polynomial system

# Algebraic Attacks

Solve for  $\mathbf{e}_1, \dots, \mathbf{e}_q$  in a polynomial system

Adapted [BØ23]'s attack to use SSD's additional structure

# Algebraic Attacks

Solve for  $\mathbf{e}_1, \dots, \mathbf{e}_q$  in a polynomial system

Adapted [BØ23]'s attack to use SSD's additional structure

Bounds on the running time of XL algorithm

# Algebraic Attacks

Solve for  $\mathbf{e}_1, \dots, \mathbf{e}_q$  in a polynomial system

Adapted [BØ23]'s attack to use SSD's additional structure

Bounds on the running time of XL algorithm

We do not find  $q > 1$  reduces security (for PCG parameters)

Not competitive with linear attacks

# Polynomial System

$\mathbf{G}_1$

$\mathbf{e}_1$

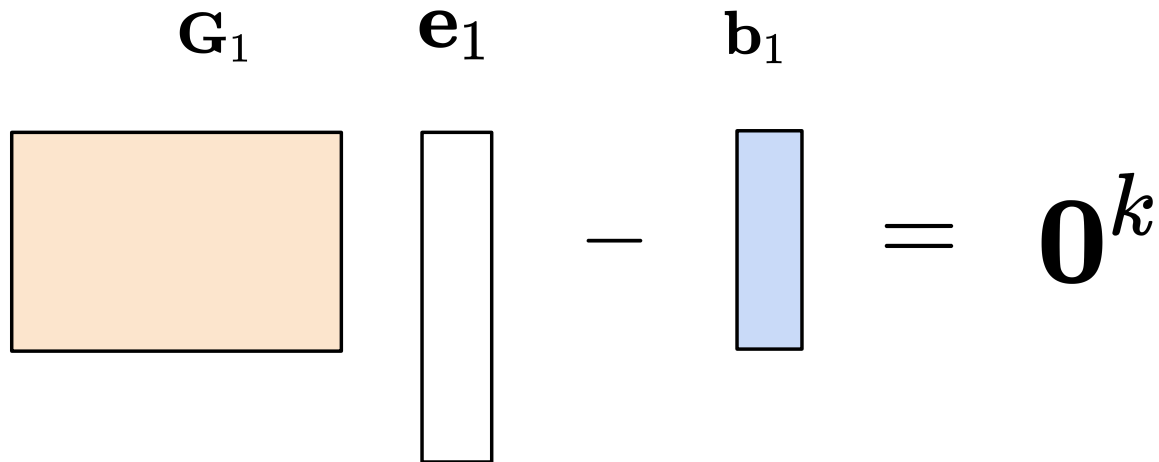
$\mathbf{b}_1$



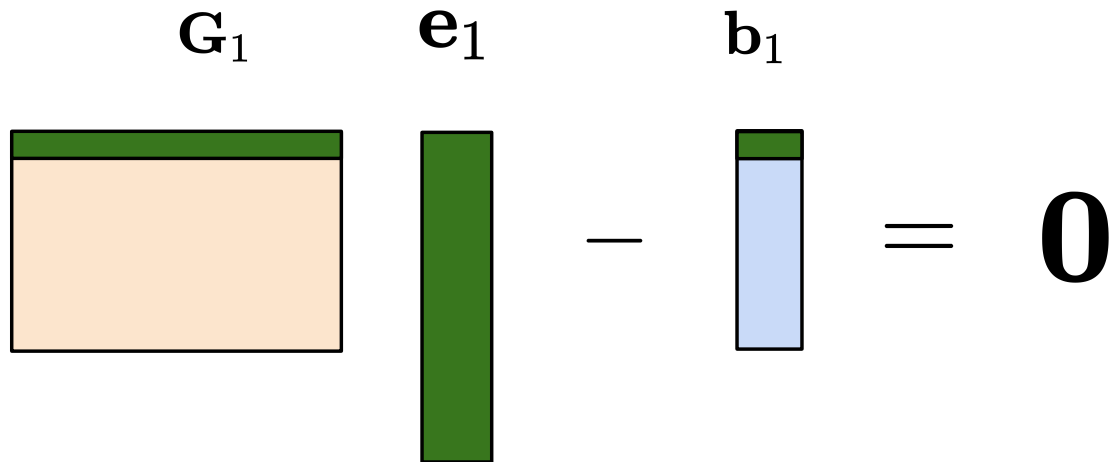
$=$



# Polynomial System

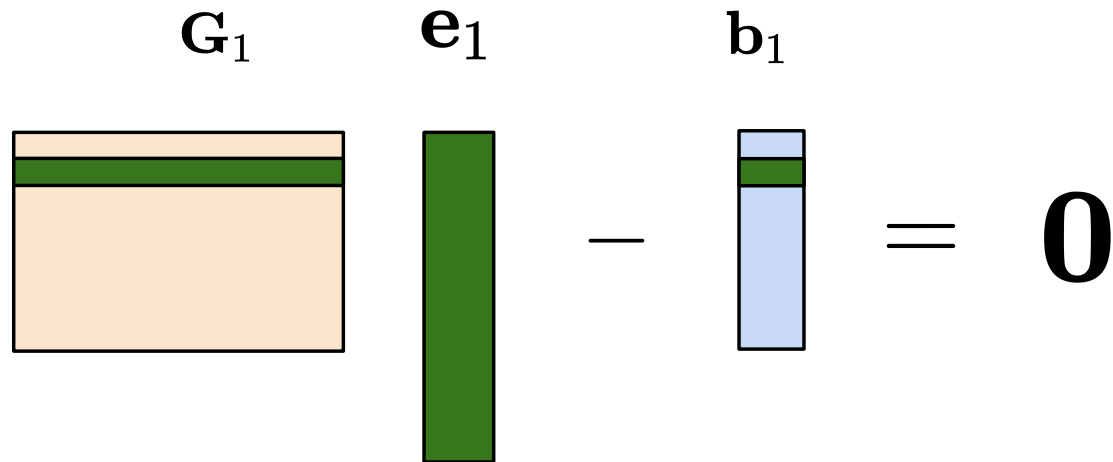
$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}^k$$
The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}^k$ . The matrix  $\mathbf{G}_1$  is represented by a wide orange rectangle. The vector  $\mathbf{e}_1$  is represented by a tall white rectangle. The vector  $\mathbf{b}_1$  is represented by a tall blue rectangle. The result  $\mathbf{0}^k$  is a bold black vector symbol.

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


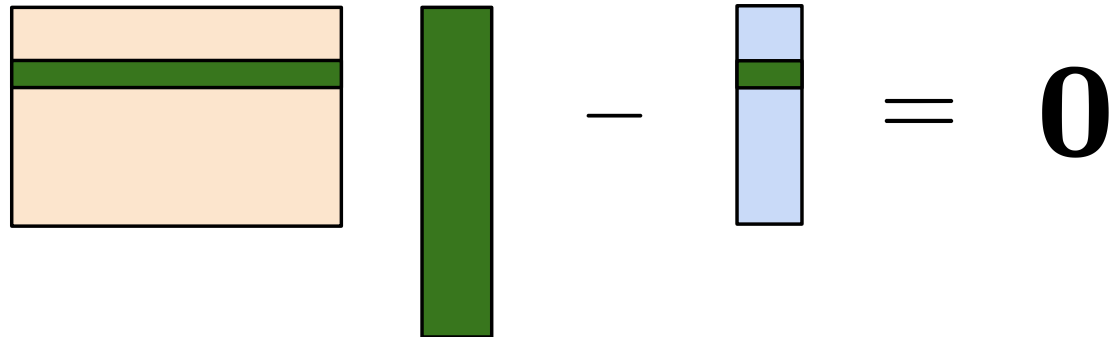
The diagram illustrates the polynomial system equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle with a green top row and orange bottom rows. The vector  $\mathbf{e}_1$  is a green column vector. The vector  $\mathbf{b}_1$  is a column vector with a green top element and a light blue bottom element. The result is the zero vector  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


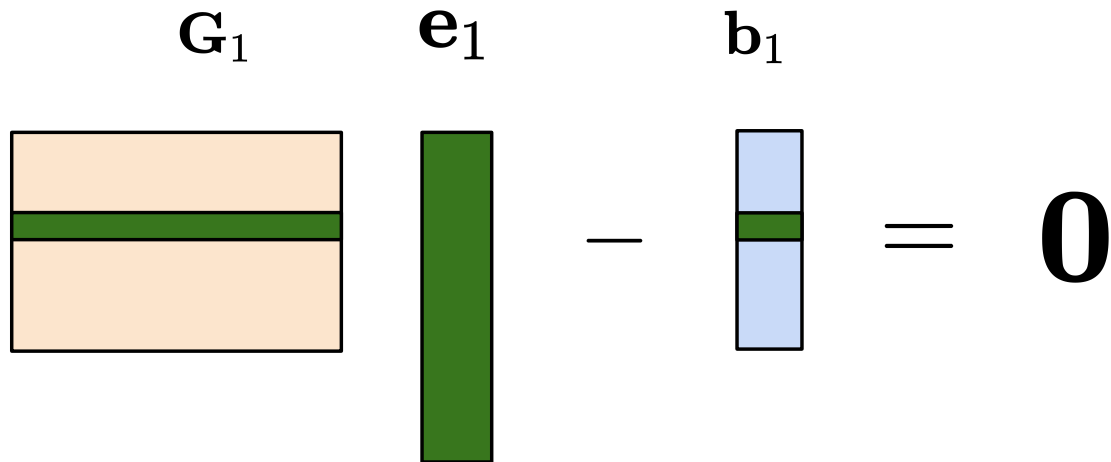
The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a wide orange rectangle with a thin green horizontal bar at the top. The vector  $\mathbf{e}_1$  is a tall green rectangle. The vector  $\mathbf{b}_1$  is a tall blue rectangle with a thin green horizontal bar at the top. The equation shows the product of  $\mathbf{G}_1$  and  $\mathbf{e}_1$  minus  $\mathbf{b}_1$  equals the zero vector, represented by a large bold  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle divided into three horizontal sections: a top orange section, a middle green section, and a bottom orange section. The vector  $\mathbf{e}_1$  is a single green column. The vector  $\mathbf{b}_1$  is a column divided into three horizontal sections: a top blue section, a middle green section, and a bottom blue section. The result is a zero vector, represented by the symbol  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle divided into three horizontal sections: two orange sections at the top and bottom, and a green section in the middle. The vector  $\mathbf{e}_1$  is a green column vector. The vector  $\mathbf{b}_1$  is a column vector divided into three horizontal sections: two light blue sections at the top and bottom, and a green section in the middle. The result is the zero vector, represented by the symbol  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$

The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle divided into three horizontal sections: two orange sections at the top and bottom, and a central green section. The vector  $\mathbf{e}_1$  is a tall green column. The vector  $\mathbf{b}_1$  is a column divided into three horizontal sections: two light blue sections at the top and bottom, and a central green section. The result is a zero vector, represented by the symbol  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$

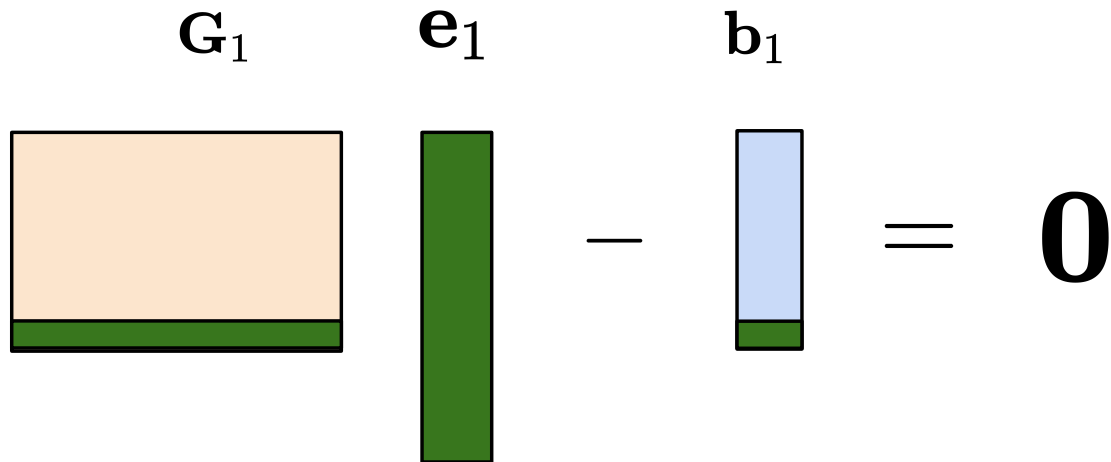
The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle with three horizontal bands: a top orange band, a middle green band, and a bottom orange band. The vector  $\mathbf{e}_1$  is a tall green rectangle. The matrix  $\mathbf{b}_1$  is a rectangle with three horizontal bands: a top light blue band, a middle green band, and a bottom light blue band. The result is the zero vector  $\mathbf{0}$ , represented by a large bold zero.

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$

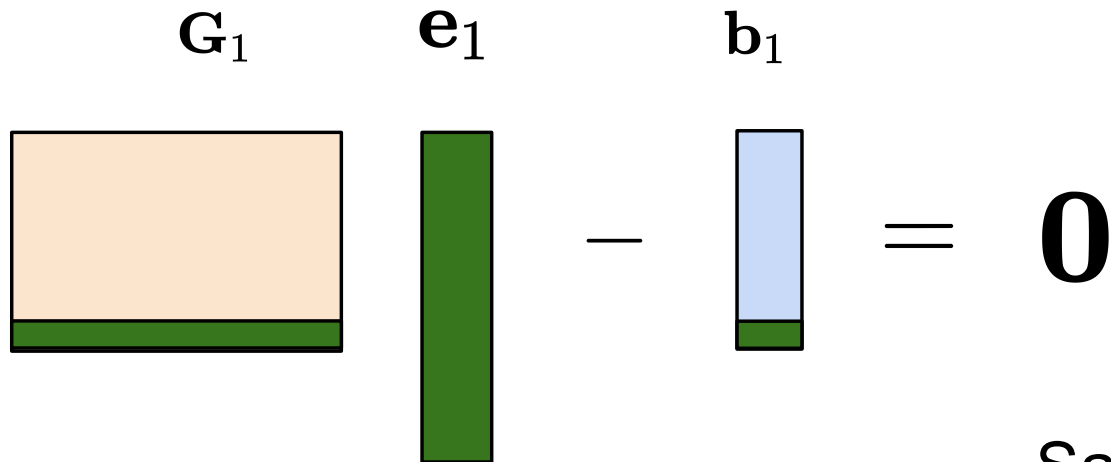
The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle divided into three horizontal sections: a large orange section at the top, a thin green section in the middle, and a thin orange section at the bottom. The vector  $\mathbf{e}_1$  is a tall green rectangle. The vector  $\mathbf{b}_1$  is a tall blue rectangle with a thin green section near the bottom. The result is a zero vector, represented by the symbol  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


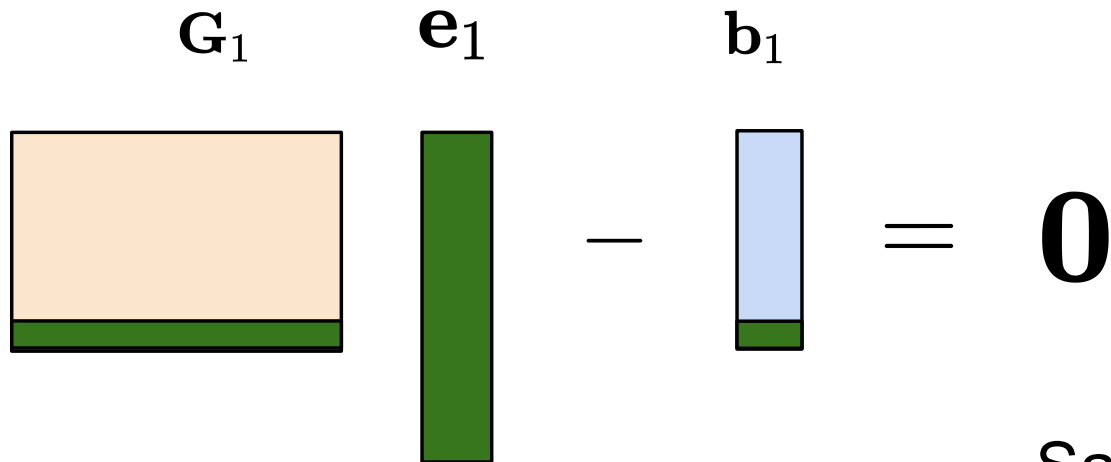
The diagram illustrates the equation  $\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$ . The matrix  $\mathbf{G}_1$  is represented by a rectangle with a light orange top and a dark green bottom. The vector  $\mathbf{e}_1$  is represented by a tall dark green rectangle. The vector  $\mathbf{b}_1$  is represented by a light blue rectangle with a dark green bottom. The equation shows the product of  $\mathbf{G}_1$  and  $\mathbf{e}_1$  minus  $\mathbf{b}_1$  equals the zero vector  $\mathbf{0}$ .

# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


Same for  $i = 2, \dots, q$

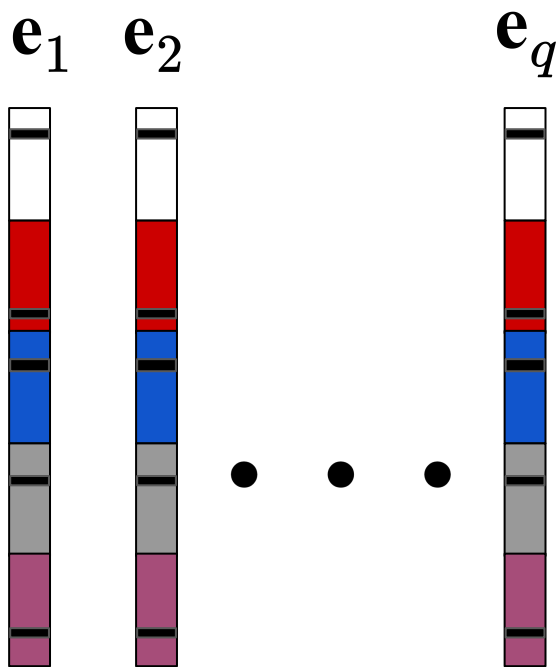
# Polynomial System

$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$


Same for  $i = 2, \dots, q$

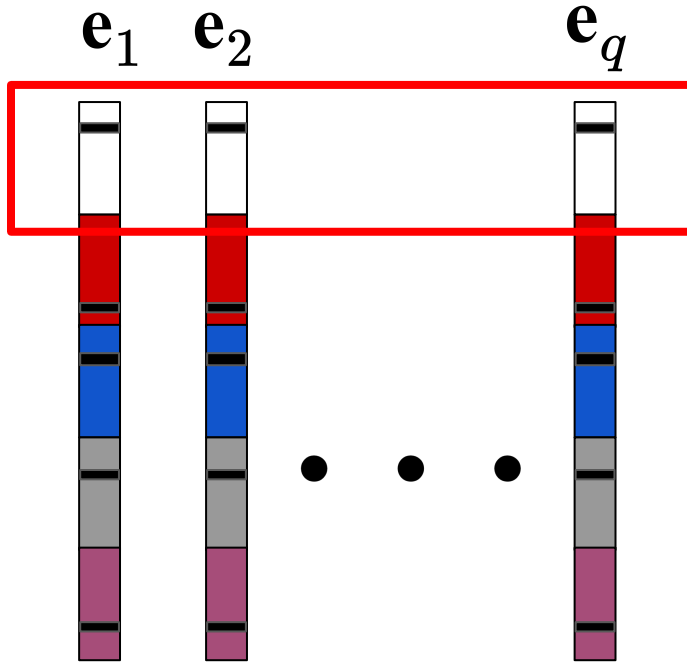
Linear

# Polynomial System



Encode stationary structure

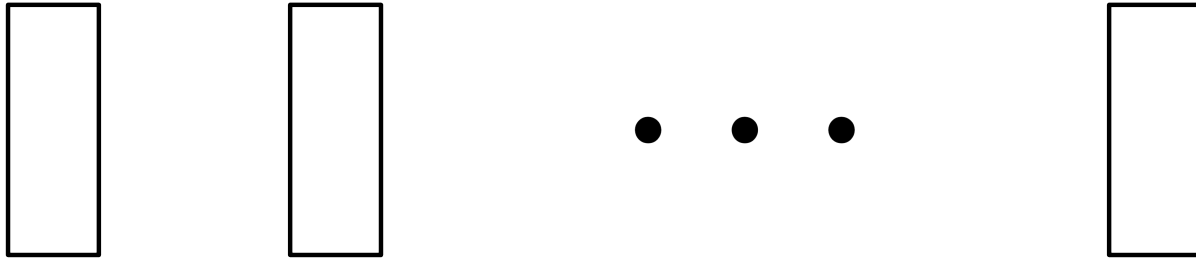
# Polynomial System



Encode stationary structure

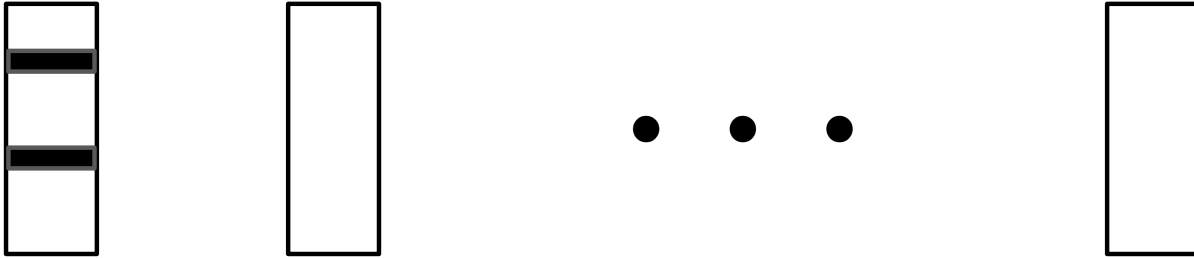
# Polynomial System

Can we multiply 2 elements in the blocks such that their output is 0?



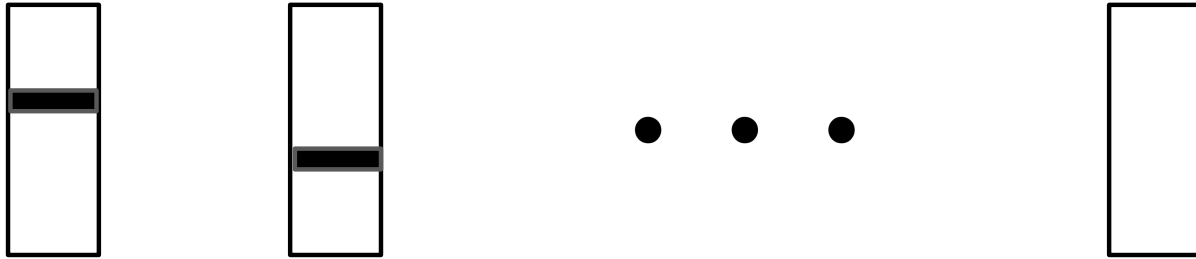
# Polynomial System

Can we multiply 2 elements in the blocks such that their output is 0?



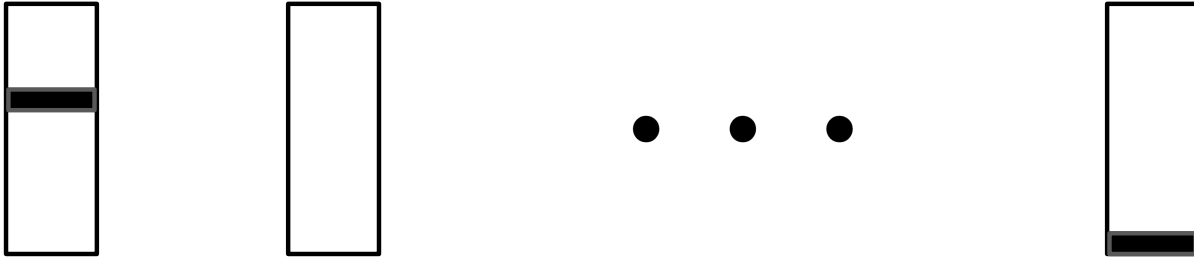
# Polynomial System

Can we multiply 2 elements in the blocks such that their output is 0?



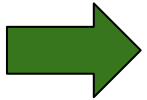
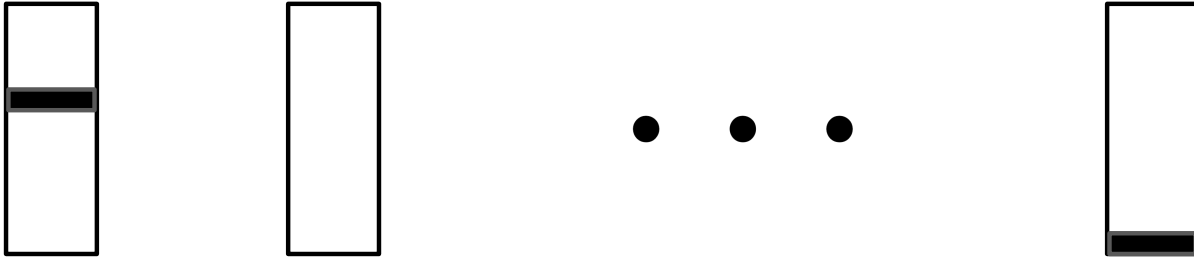
# Polynomial System

Can we multiply 2 elements in the blocks such that their output is 0?



# Polynomial System

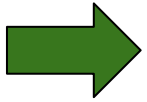
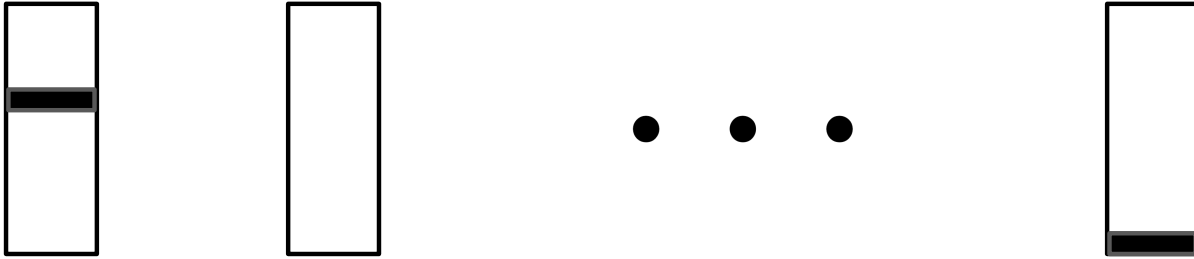
Can we multiply 2 elements in the blocks such that their output is 0?



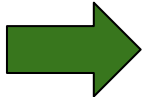
They just cannot be in the same row

# Polynomial System

Can we multiply 2 elements in the blocks such that their output is 0?



They just cannot be in the same row



Quadratic

# Polynomial System

In  $\mathbb{F}_2$  , we also add field equations

# High-Level Approach

Construct the system of polynomials  $F = \{f_1, \dots, f_p\}$

Apply the XL Algorithm [CKPS00]

# High-Level Approach

Construct the system of polynomials  $F = \{f_1, \dots, f_p\}$

Apply the XL Algorithm [CKPS00]

1. Map the non-linear system to a linear system
2. Solve using standard techniques (Gaussian elimination)

# High-Level Approach

Construct the system of polynomials  $F = \{f_1, \dots, f_p\}$

Apply the XL Algorithm [CKPS00]

1. Map the non-linear system to a linear system
  - a. Multiply each  $f_i$  by arbitrary monomials so the resulting polynomials are of degree  $\leq d$
2. Solve using standard techniques (Gaussian elimination)

# High-Level Approach

Construct the system of polynomials  $F = \{f_1, \dots, f_p\}$

Apply the XL Algorithm [CKPS00]

1. Map the non-linear system to a linear system
  - a. Multiply each  $f_i$  by arbitrary monomials so the resulting polynomials are of degree  $\leq d$
  - b. Linearize  $F$  by treating its monomials as new variables and save their coefficients in the Macaulay matrix
2. Solve using standard techniques (Gaussian elimination)

# High-Level Approach

Construct the system of polynomials  $F = \{f_1, \dots, f_p\}$

Apply the XL Algorithm [CKPS00]

1. Map the non-linear system to a linear system
  - a. Multiply each  $f_i$  by arbitrary monomials so the resulting polynomials are of degree  $\leq d$

**Witness degree**

- b. Linearize  $F$  by treating its monomials as new variables and save their coefficients in the Macaulay matrix
2. Solve using standard techniques (Gaussian elimination)

# Witness Degree

For XL to succeed we need to produce enough new equations

# Witness Degree

For XL to succeed we need to produce enough new equations  
d determines:

- Size of Macaulay matrix

- Cost of Gaussian elimination

- Key cost of XL

# Witness Degree

For XL to succeed we need to produce enough new equations  
d determines:

- Size of Macaulay matrix

- Cost of Gaussian elimination

- Key cost of XL

 Computing d is the key challenge (from Hilbert series)

# Experimental Evaluation

Implemented OT and VOLE from SD/SSD

Reduce communication 6-18x

Reduce runtime 1.5x

# Work in Submission

Our new work significantly accelerates multiplication by **G**

Thus, the cost of generating  $[\Delta e]$  becomes even more significant

# Work in Submission

Our new work significantly accelerates multiplication by **G**

Thus, the cost of generating  $[\Delta e]$  becomes even more significant

Another work closely relies on SSD to generate Beaver triples

# Stationary Syndrome Decoding (SSD)

Allows reusing noisy coordinates of  $\mathbf{e}$  across  $q$  SD instances

Significant impact on PCG Performance

# Stationary Syndrome Decoding (SSD)

Allows reusing noisy coordinates of  $\mathbf{e}$  across  $q$  SD instances

Significant impact on PCG Performance

Excited to see novel applications of SSD

We invite the community to analyze SSD and its variants