Stationary Syndrome Decoding for Improved PCGs

Stan Peceny (Now at Stealth Software Technologies)

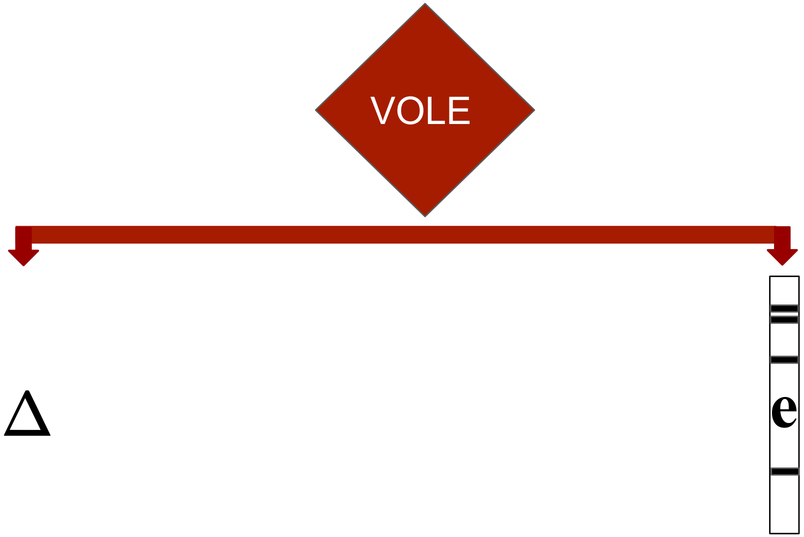
Joint work with:
Vlad Kolesnikov, Srini Raghuraman, Peter Rindal

Georgia Institute of Technology

VISA

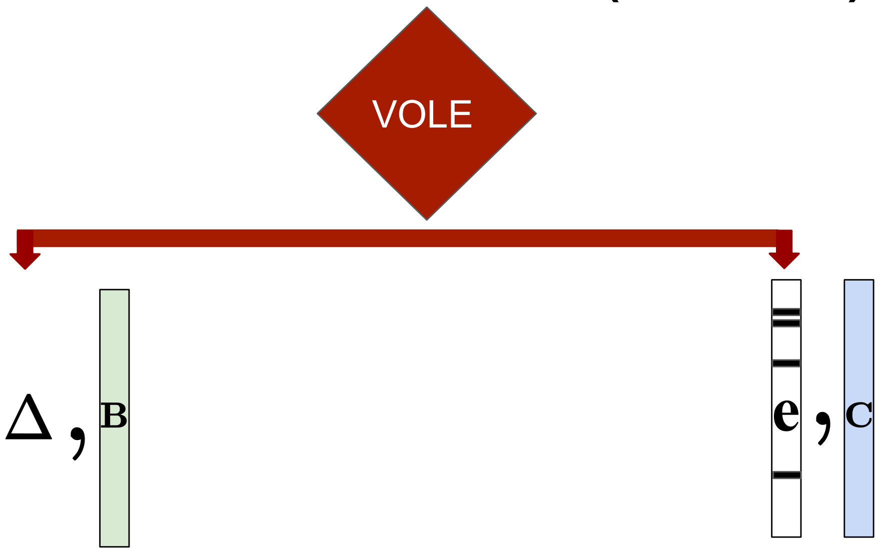
Vector Oblivious Linear Evaluation (VOLE)

$[\Delta\mathbf{e}]$



Vector Oblivious Linear Evaluation (VOLE)

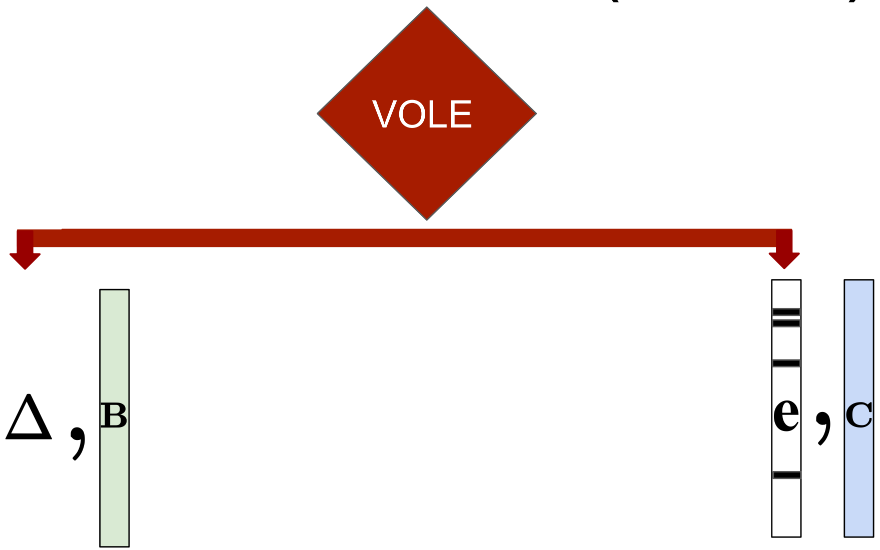
$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$



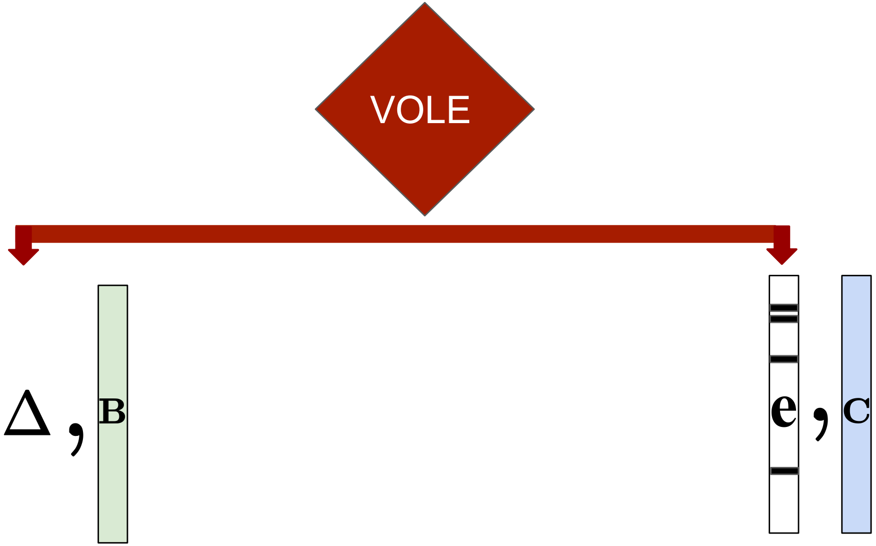
Vector Oblivious Linear Evaluation (VOLE)

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

Interactive (sublinear comm.)



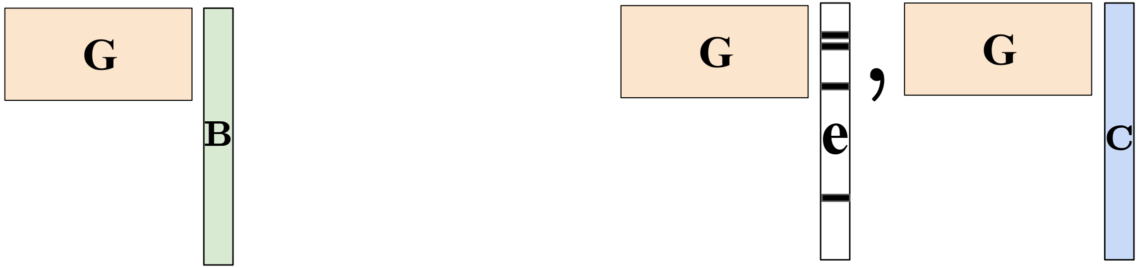
Vector Oblivious Linear Evaluation (VOLE)



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

Interactive (sublinear comm.)

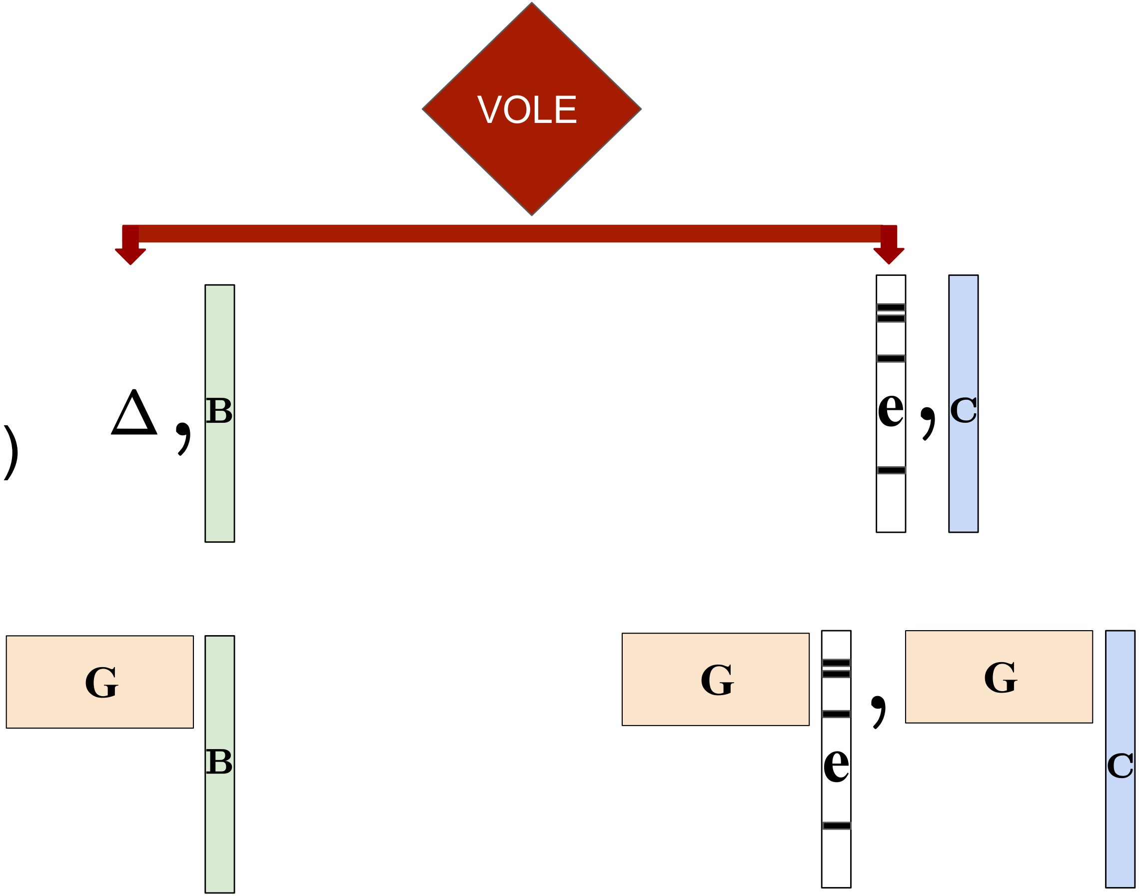
$$[\mathbf{G}\Delta \mathbf{e}]$$



Vector Oblivious Linear Evaluation (VOLE)

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

Interactive (sublinear comm.)



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

Local

Vector Oblivious Linear Evaluation (VOLE)

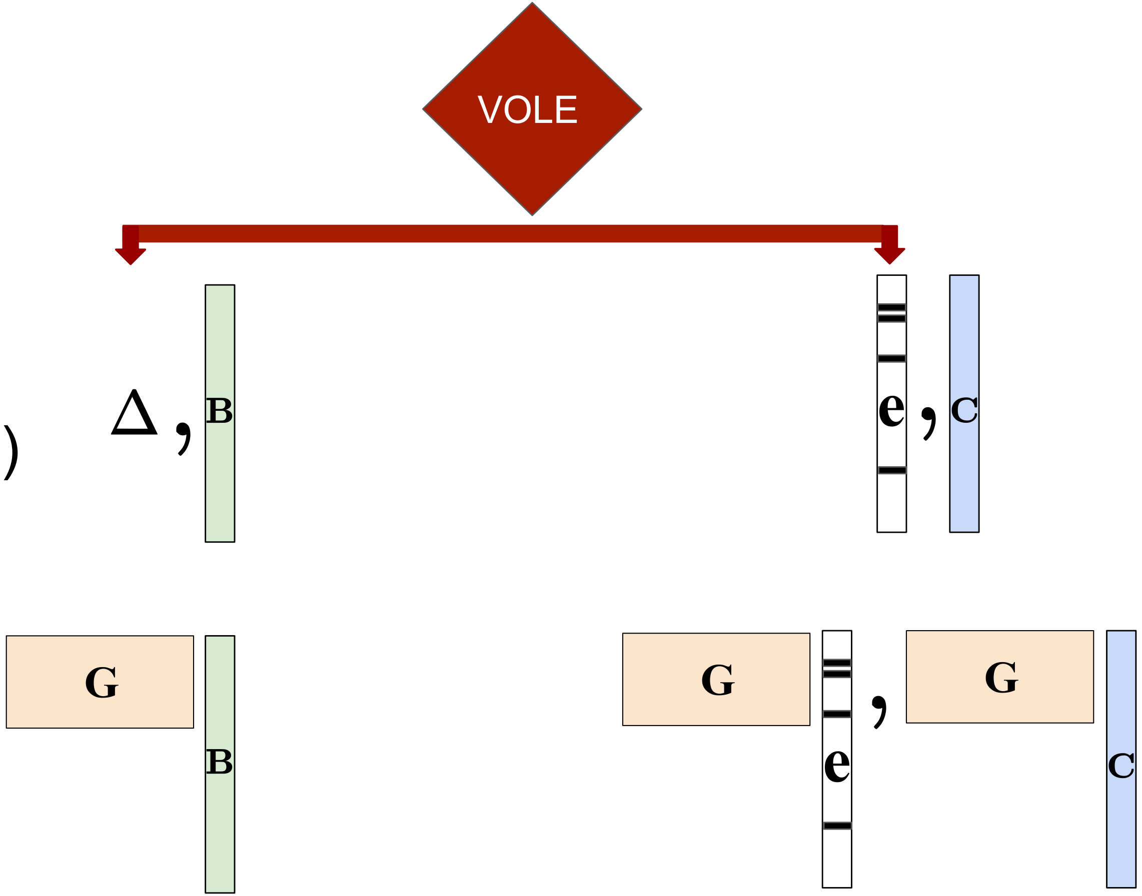
Similar for OT

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

Interactive (sublinear comm.)

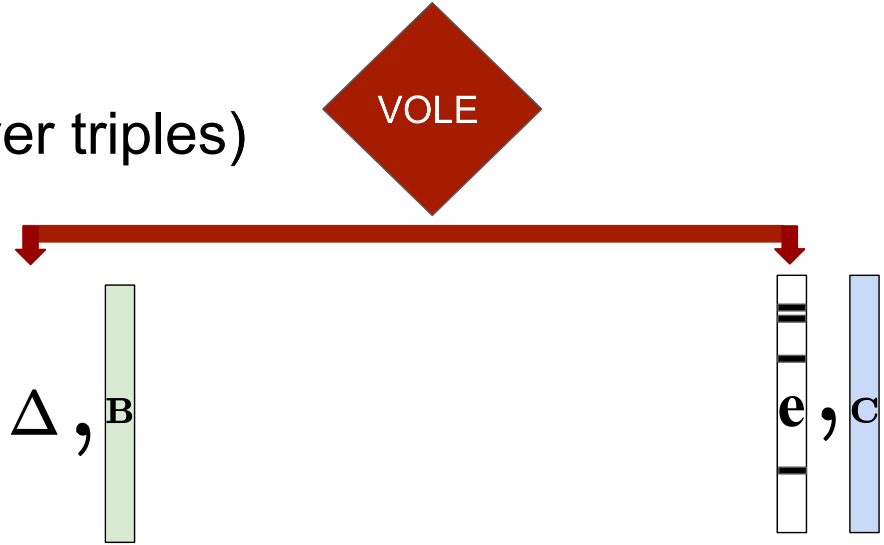
$$[\mathbf{G}\Delta\mathbf{e}]$$

Local



Vector Oblivious Linear Evaluation (VOLE)

Similar for OT
Other correlations (Beaver triples)

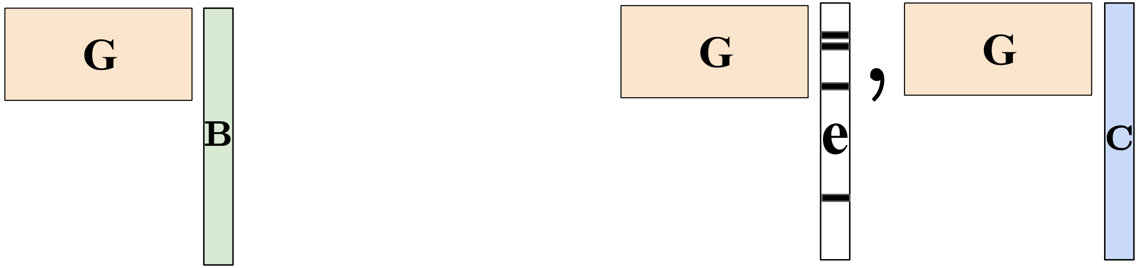


$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$

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

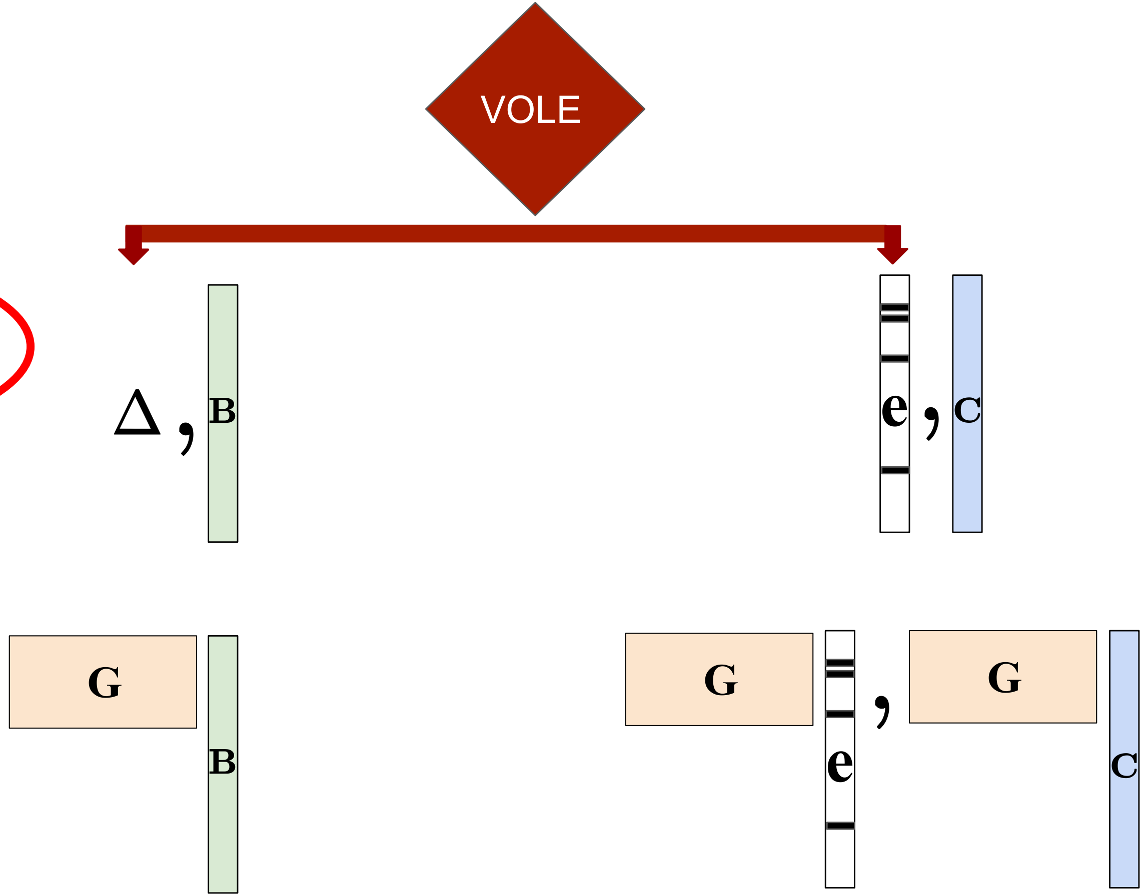
State of the art for generating correlated randomness

Correlated randomness is essential for MPC

Vector Oblivious Linear Evaluation (VOLE)

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

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

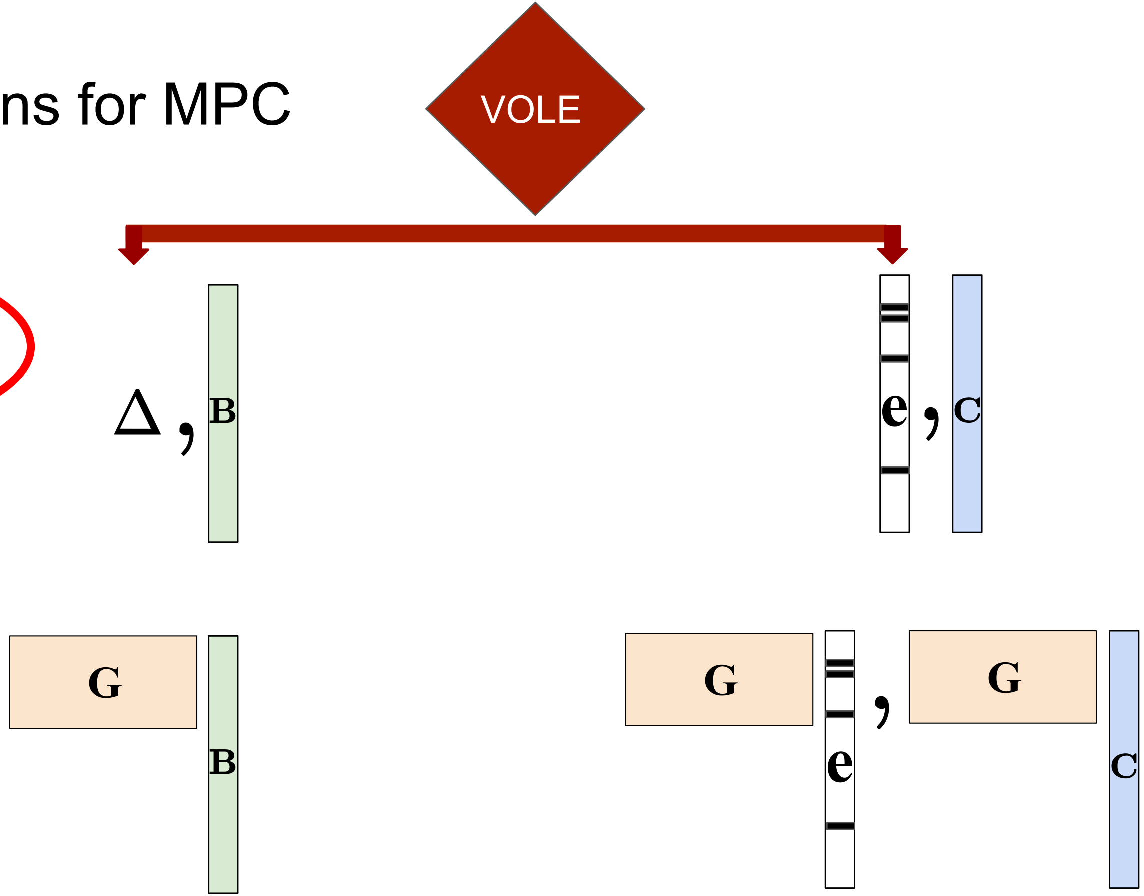


Vector Oblivious Linear Evaluation (VOLE)

- Need many correlations for MPC

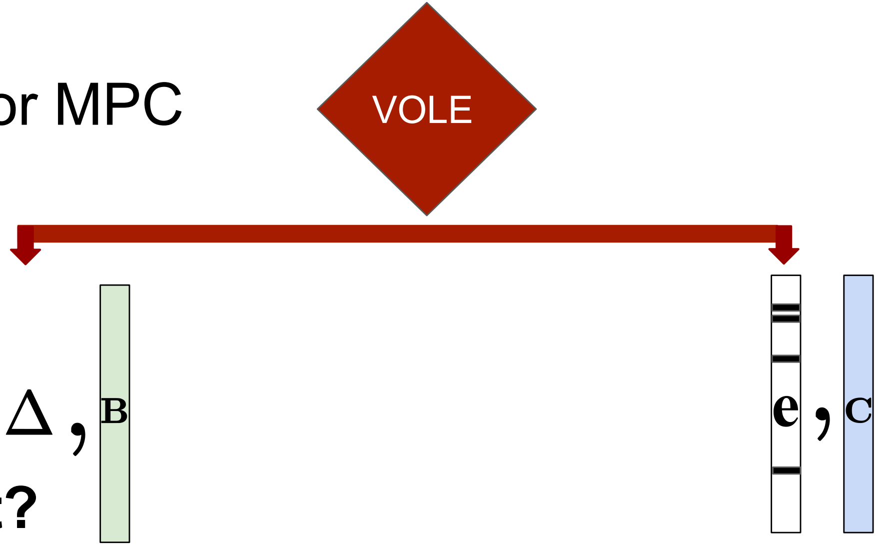
$$[\Delta \mathbf{e}] = \mathbf{B} + \mathbf{C}$$

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



Vector Oblivious Linear Evaluation (VOLE)

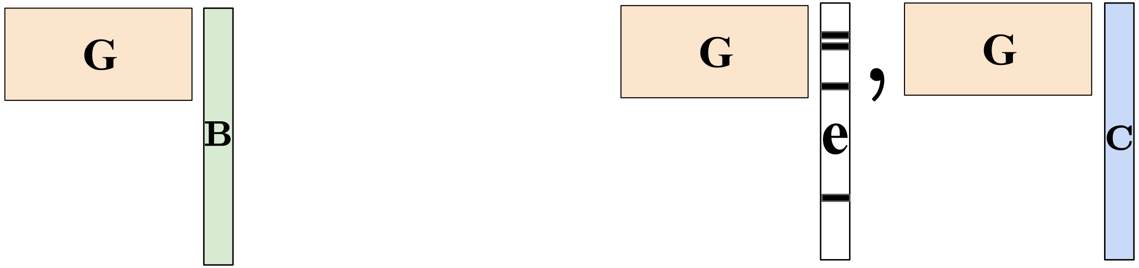
➡ Need many correlations for MPC



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

Can we amortize this cost?

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

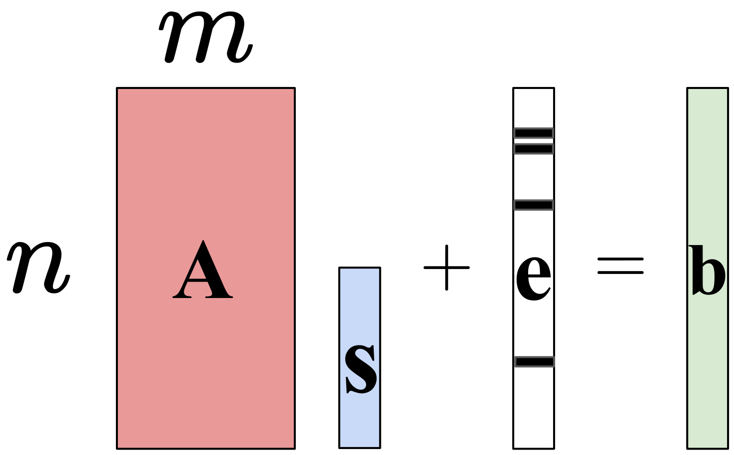


LPN

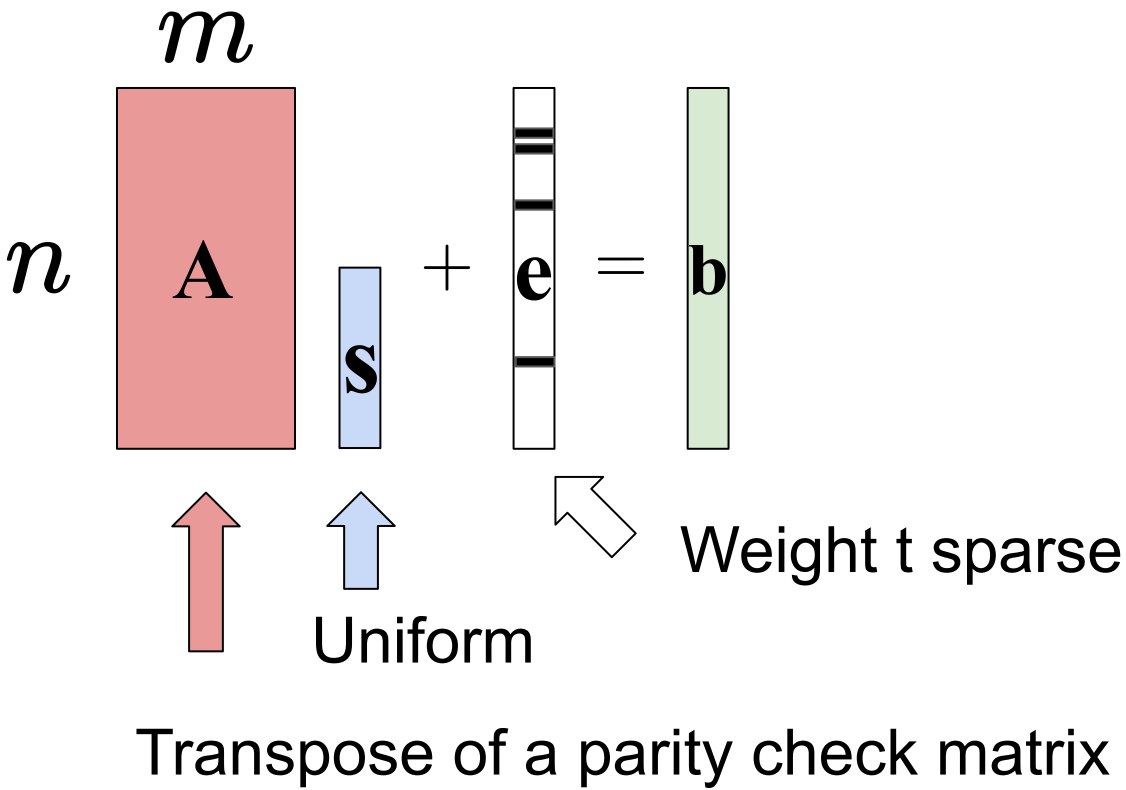
Syndrome Decoding (SD)

LPN

Syndrome Decoding (SD)

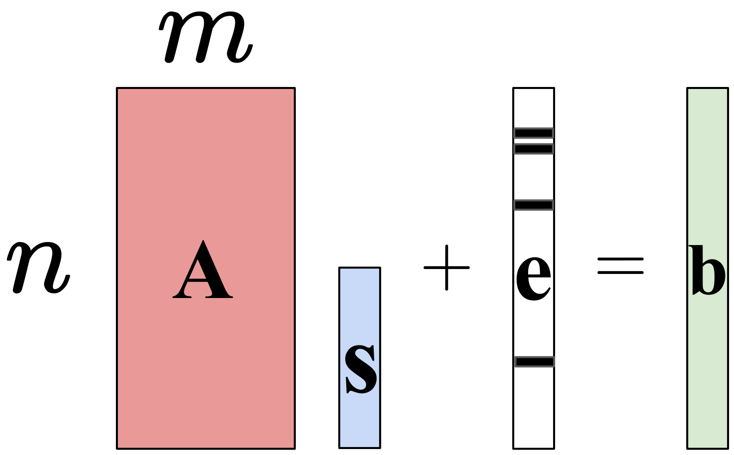


LPN | Syndrome Decoding (SD)



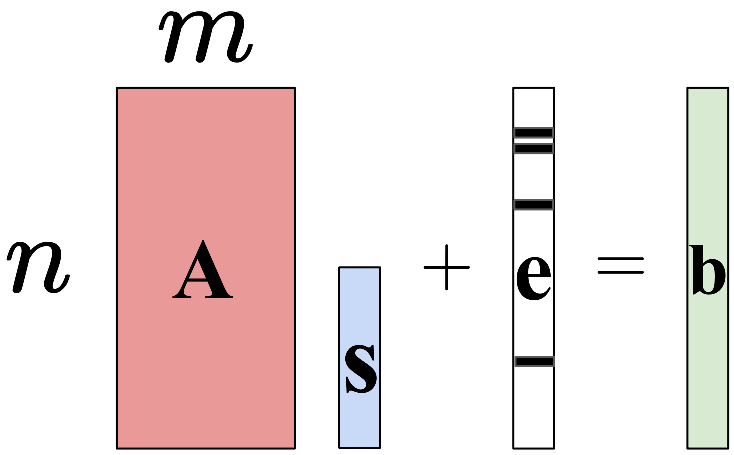
LPN

Syndrome Decoding (SD)



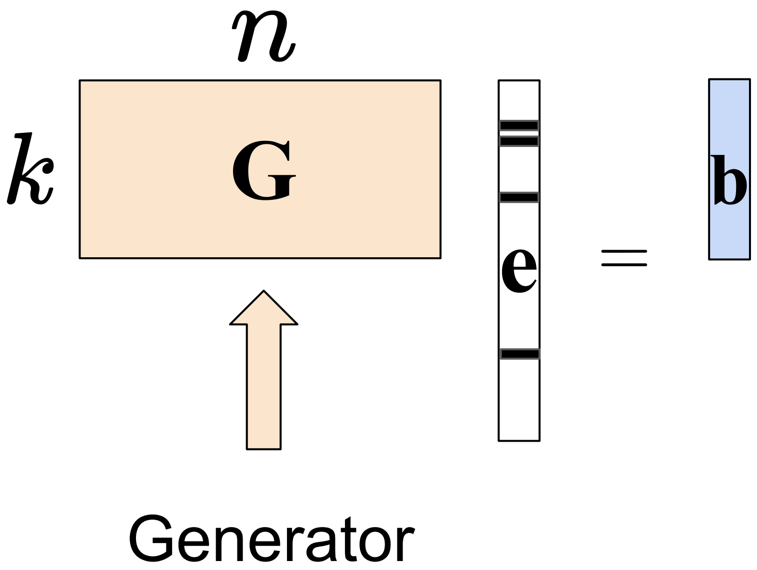
$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

LPN

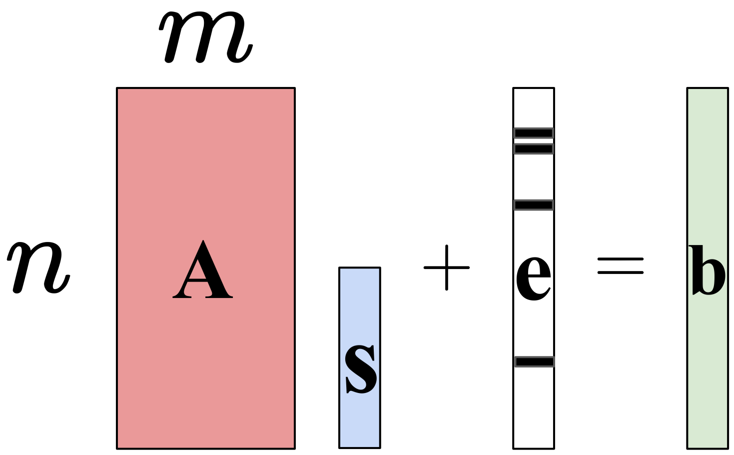


$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

Syndrome Decoding (SD)

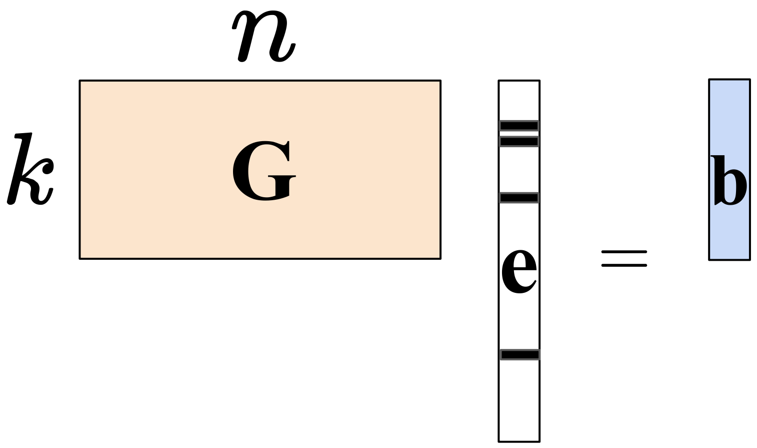


LPN



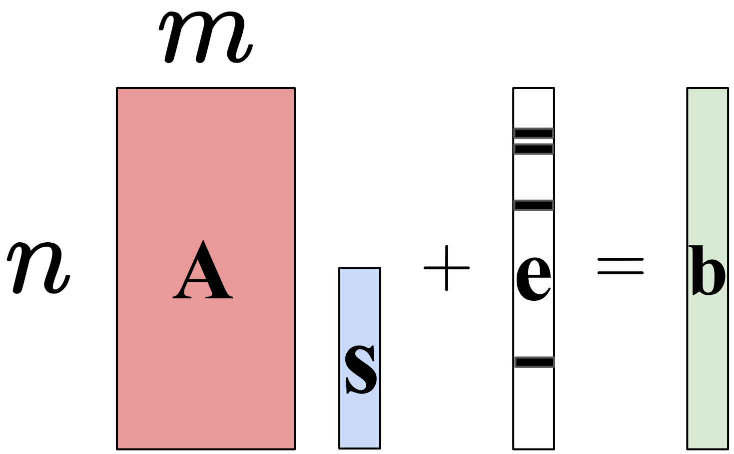
$$(\mathbf{A}, \mathbf{b}) \approx (\mathbf{A}, \$)$$

Syndrome Decoding (SD)



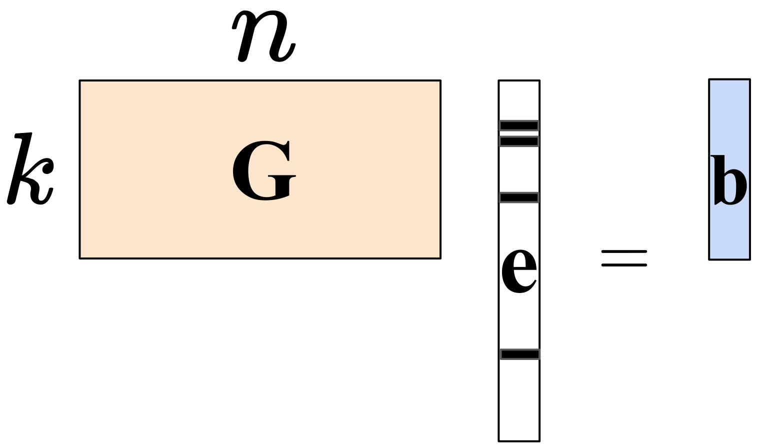
$$(\mathbf{G}, \mathbf{b}) \approx (\mathbf{G}, \$)$$

LPN



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

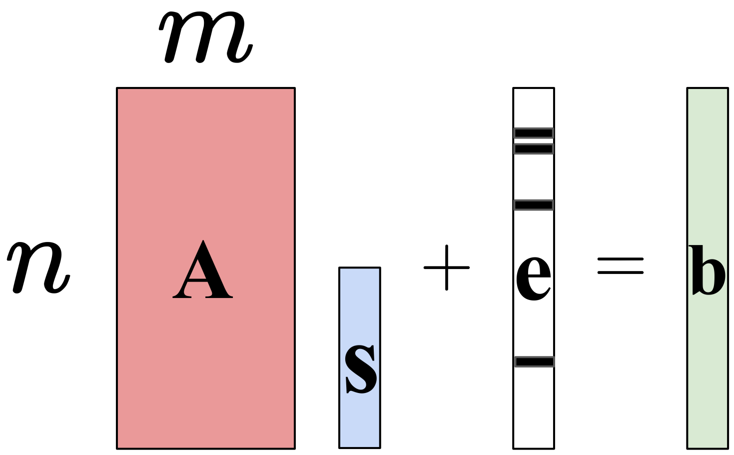
Syndrome Decoding (SD)



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

LPN and SD are equivalent

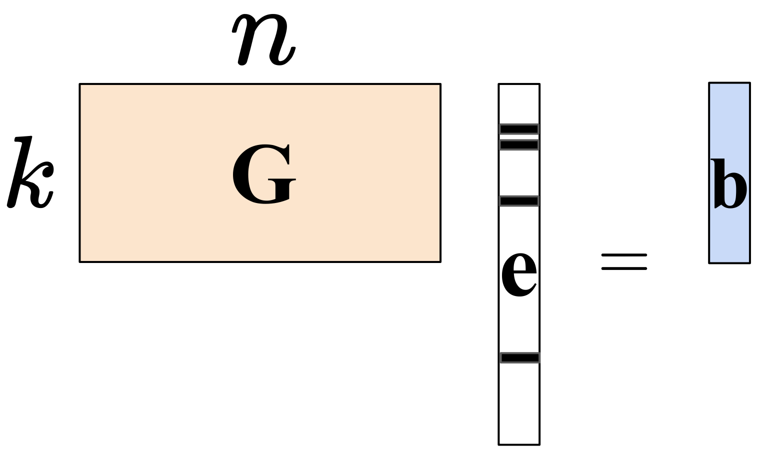
LPN



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

LPN and SD are equivalent

Syndrome Decoding (SD)



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

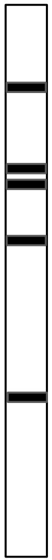
Used for PCGs

Syndrome Decoding (SD)

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

Noise Distributions

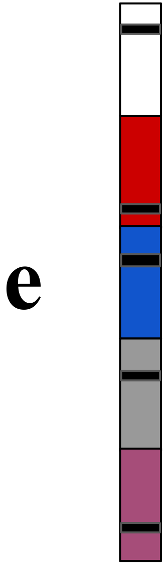
e



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

Exact - fixes Hamming weight to t

Noise Distributions

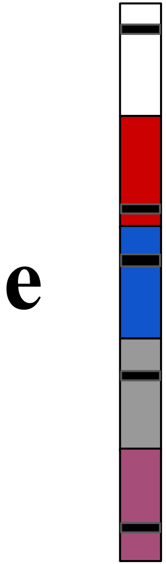


Bernoulli - classic, sample $\mathbf{e}_i$ with $\mathrm{Ber}_{t/n}$

Exact - fixes Hamming weight to t

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

Noise Distributions



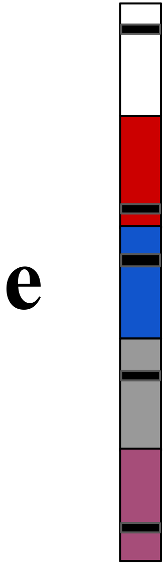
Bernoulli - classic, sample **$\mathbf{e}_i$** with $\mathrm{Ber}_{t/n}$

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



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

Exact - fixes Hamming weight to t

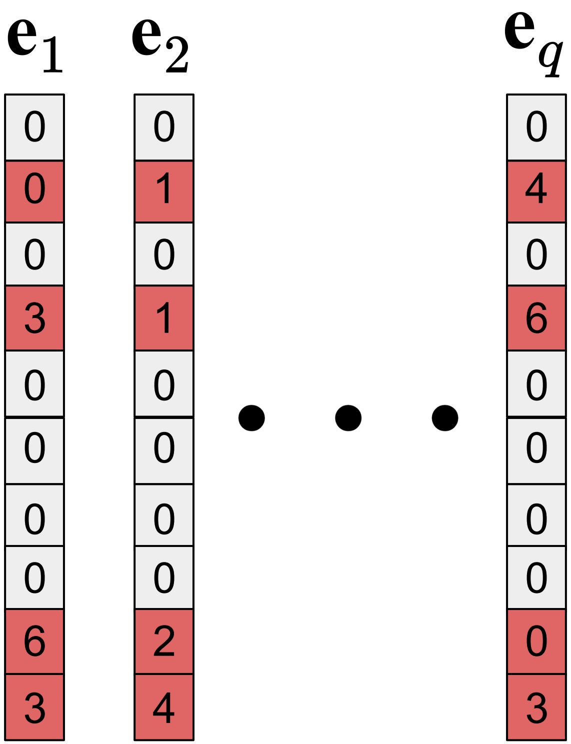
Regular - t same-size blocks, each a random unit vector

All of these improve for 1 instance

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

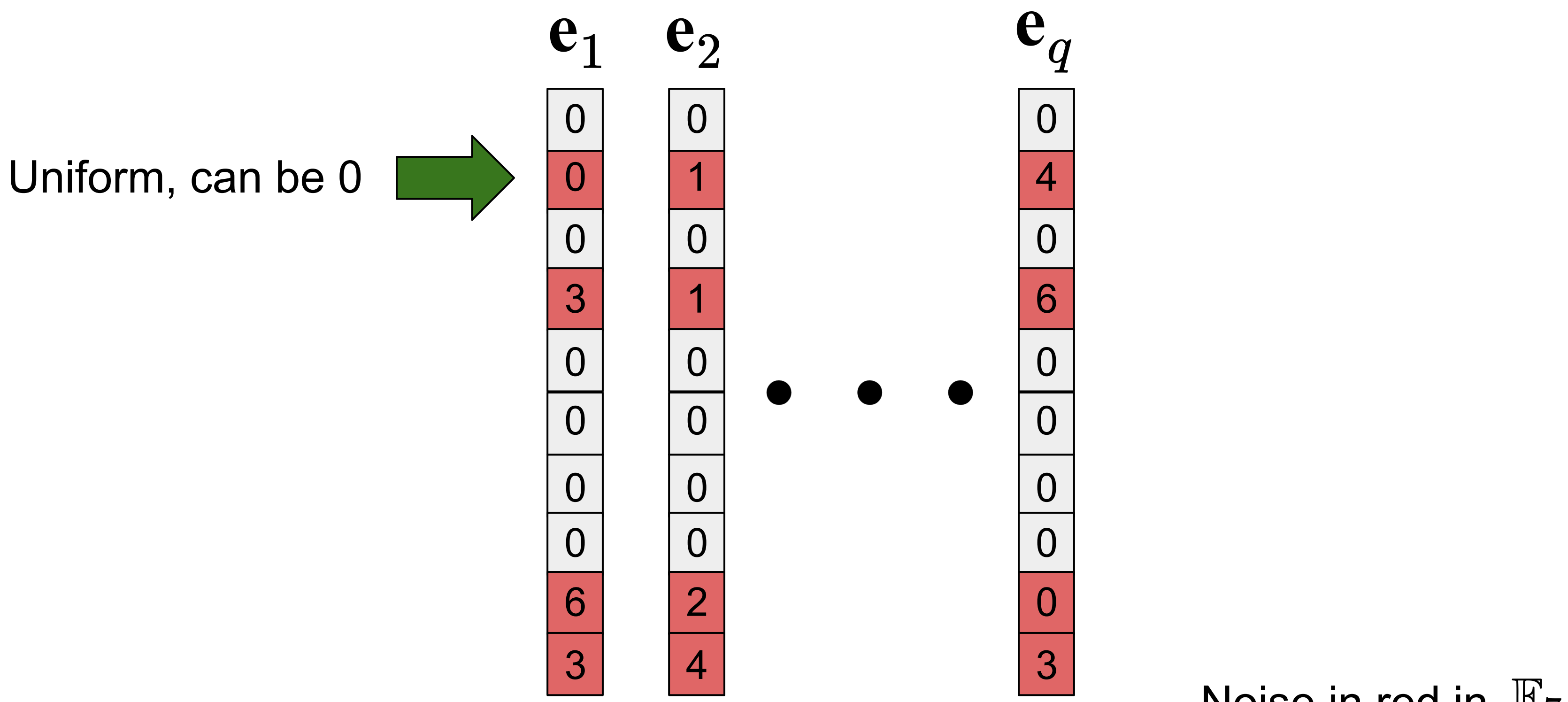
Stationary Syndrome Decoding (SSD)

Noisy coordinates reused

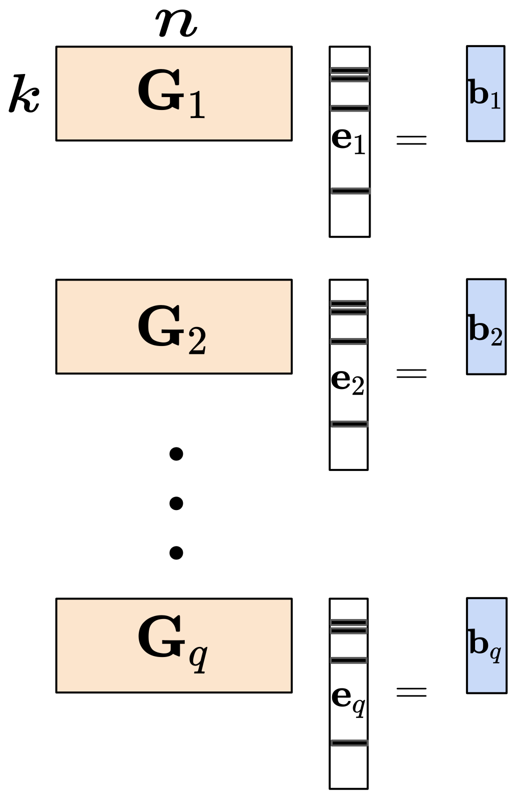


Noise in red in $\mathbb{F}_7$

Stationary Syndrome Decoding (SSD)

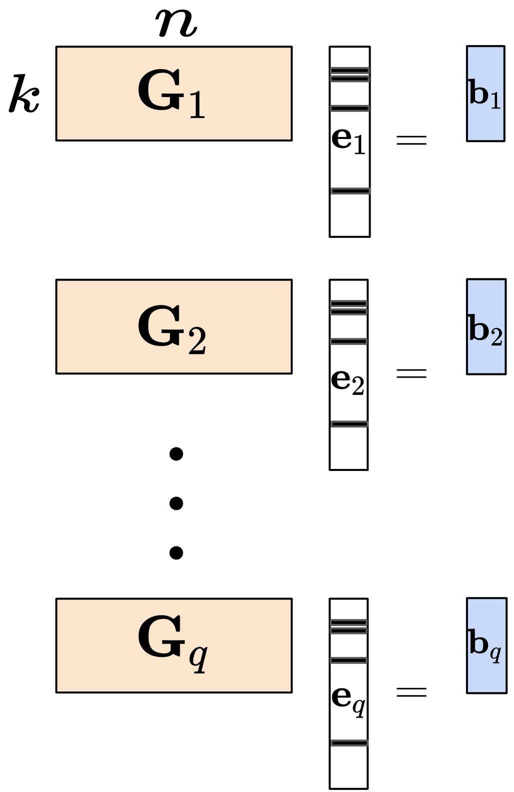


Stationary Syndrome Decoding



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

Stationary Syndrome Decoding



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

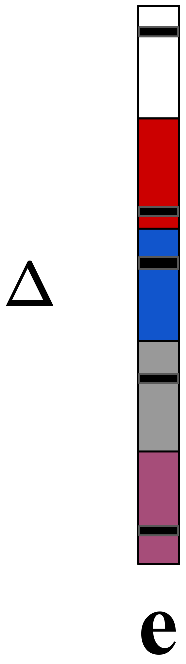
We define SLPN similarly

Stationary Syndrome Decoding (SSD)

We cryptanalyze for $\mathbf{G}_i$

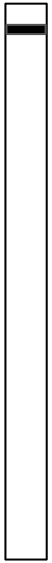
with high minimum distance and regular $\mathbf{e}_i$

How Does SSD Help with PCGs?

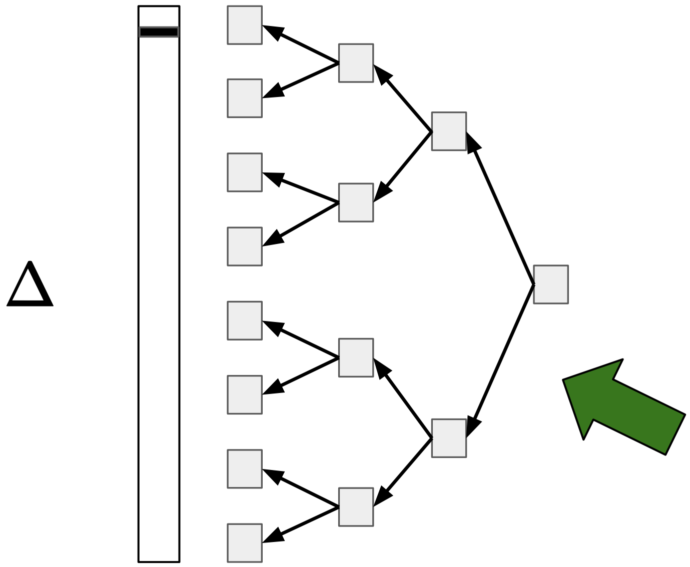


How Does SSD Help with PCGs?

Δ

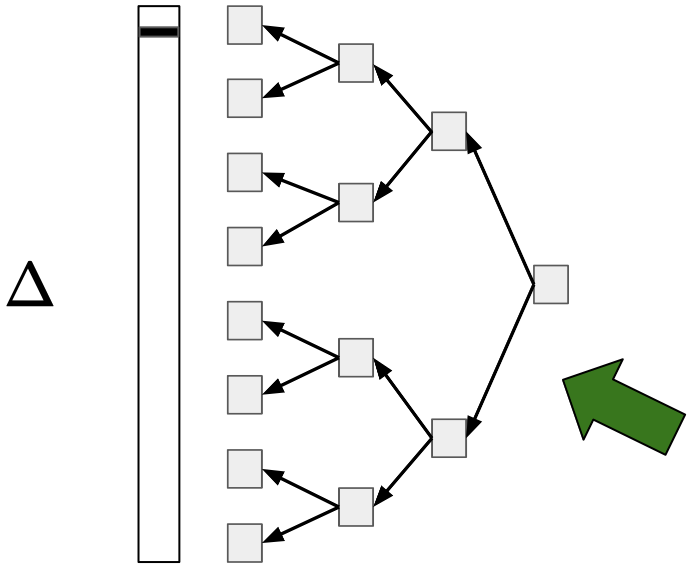


How Does SSD Help with PCGs?



OT at each level

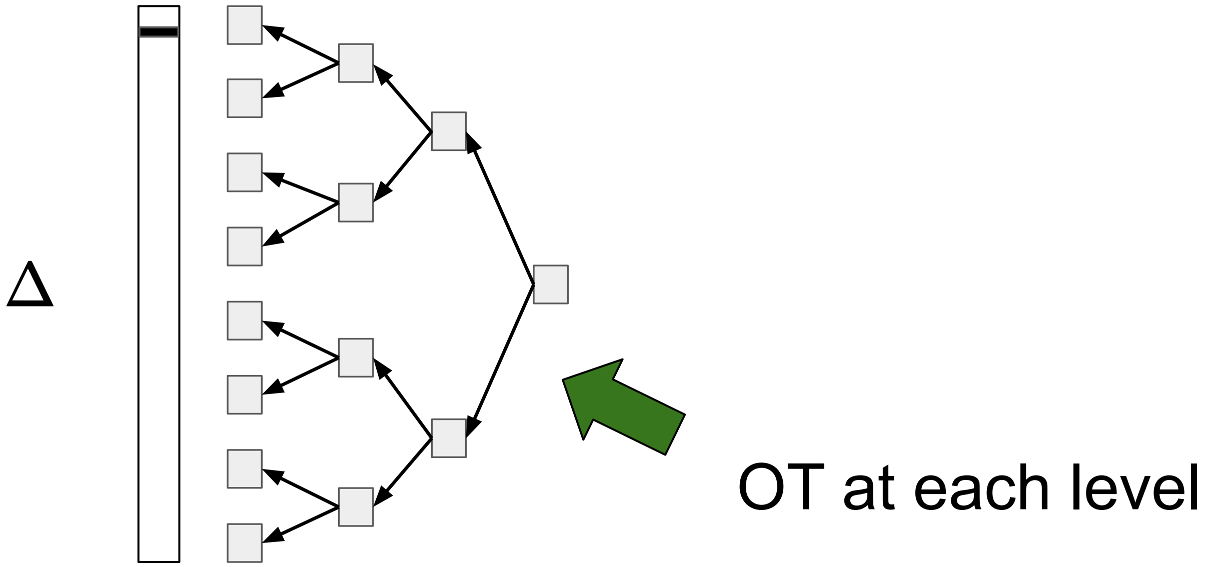
How Does SSD Help with PCGs?



OT at each level

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]

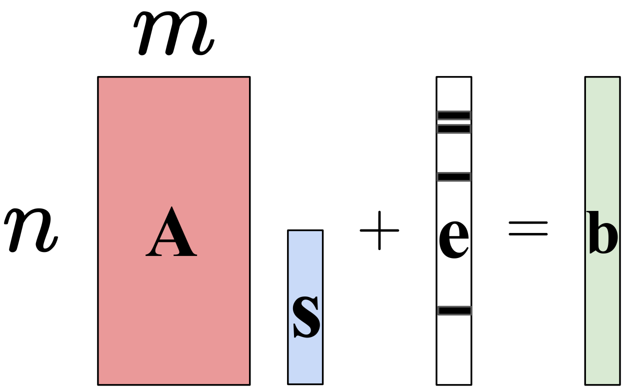
Low Weight Code [Zic17]

…

Tedious to go through each attack

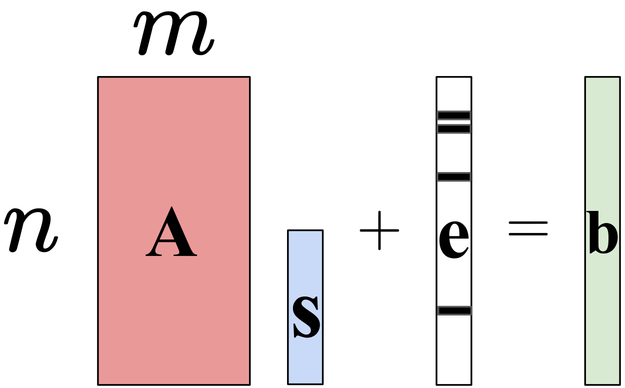
Linear Test Framework

LPN



Linear Test Framework

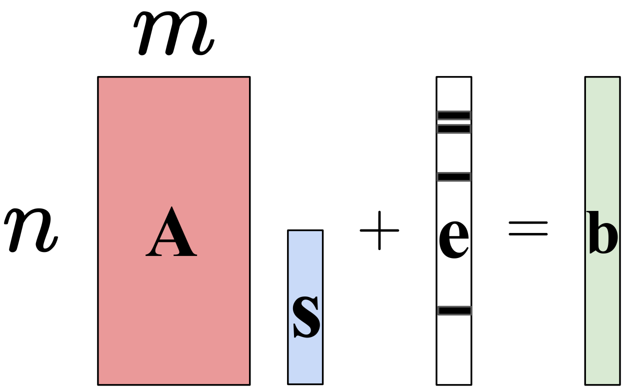
LPN

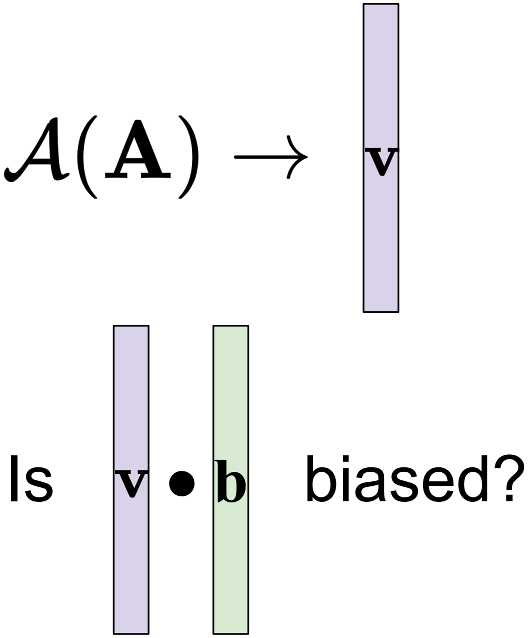


$$\mathcal{A}(\mathbf{A}) \to \mathbf{v}$$

Linear Test Framework

LPN





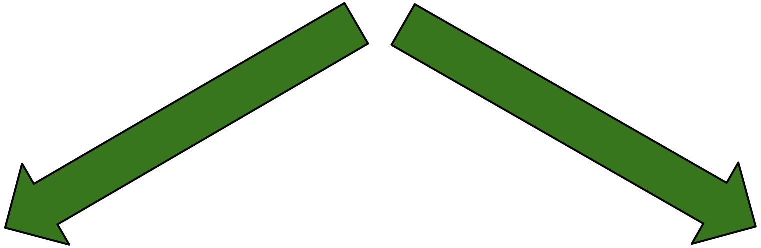
$$\mathbf{v} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e})$$

High-Level Proof Template

For SLPN with **regular** noise

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

High-Level Proof Template



$q = 1$

$q > 1$

High-Level Proof Template

$q = 1$

$q > 1$

Differs from plain LPN
with regular noise

High-Level Proof Template

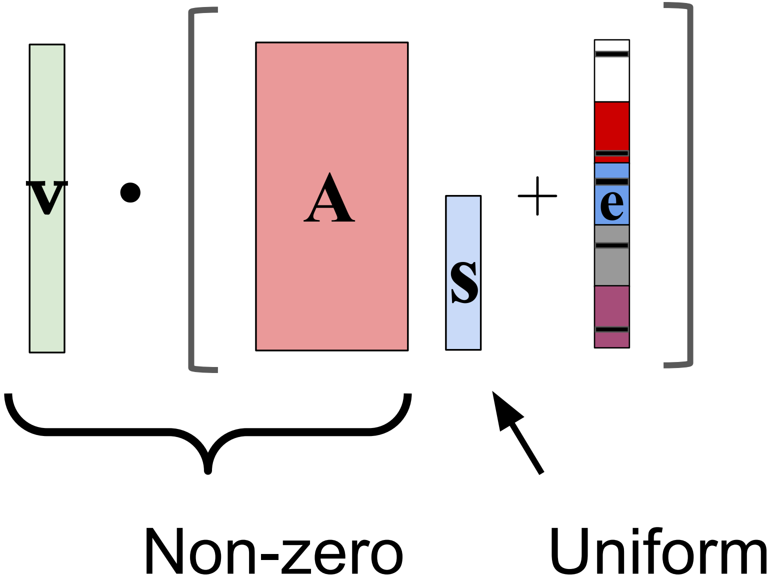
$$q = 1$$

non-codeword **v**

High-Level Proof Template

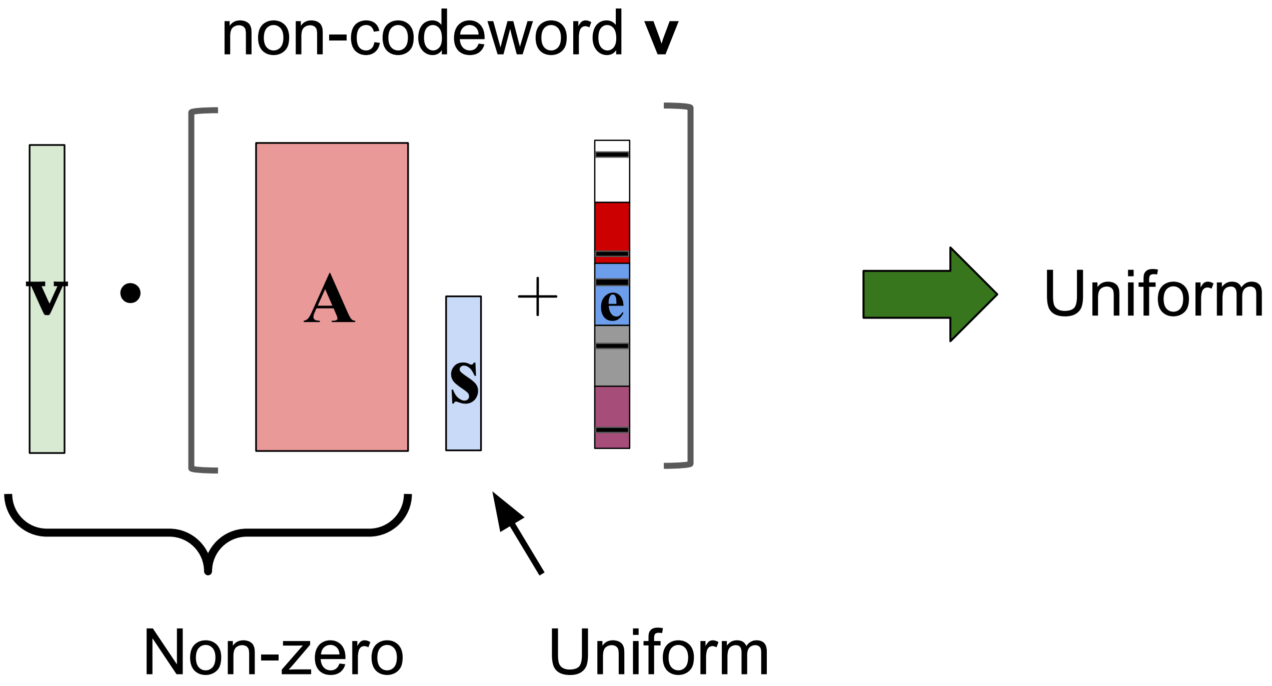
$$q = 1$$

non-codeword **v**



High-Level Proof Template

$$q = 1$$



High-Level Proof Template

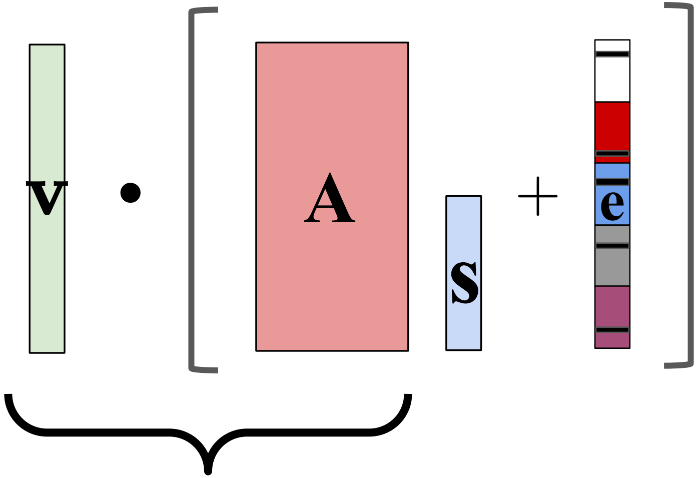
$$q = 1$$

codeword **v**

High-Level Proof Template

$$q = 1$$

codeword **v**

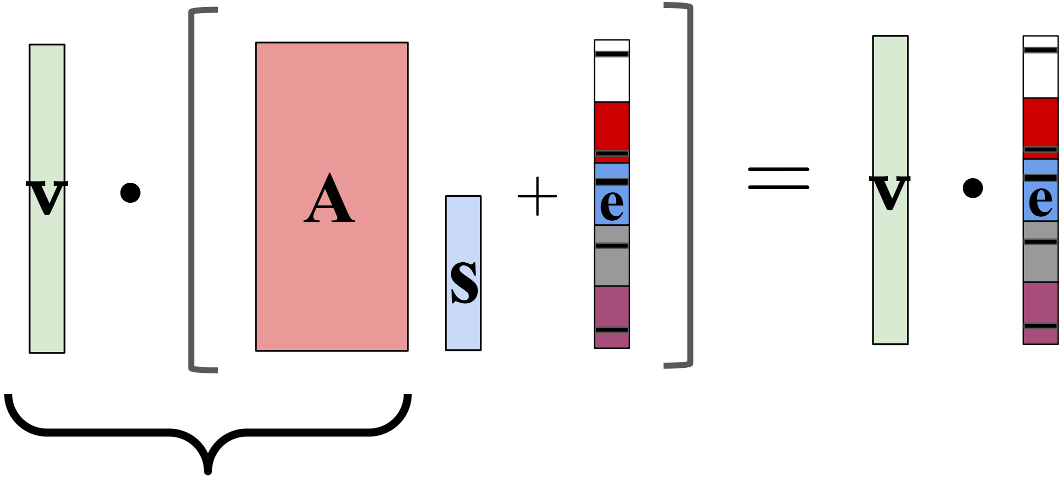


Zero, randomness by **s** vanishes

High-Level Proof Template

$$q = 1$$

codeword **v**

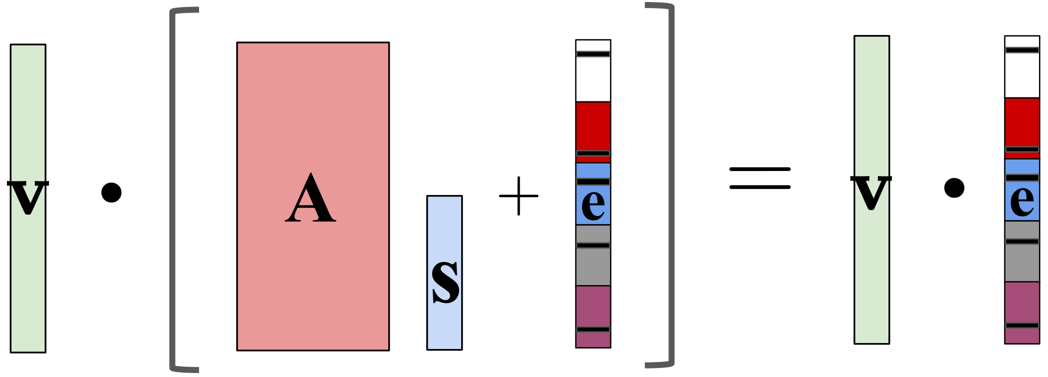


Zero, randomness by **s** vanishes

High-Level Proof Template

$$q = 1$$

codeword $\mathbf{v}$

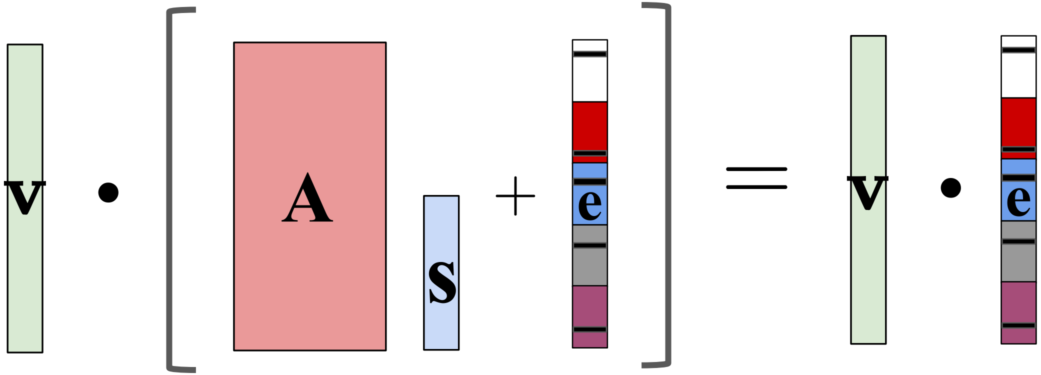


→ Need to show $\mathbf{v} \cdot \mathbf{e}$ has negligible bias

High-Level Proof Template

$$q = 1$$

codeword **v**

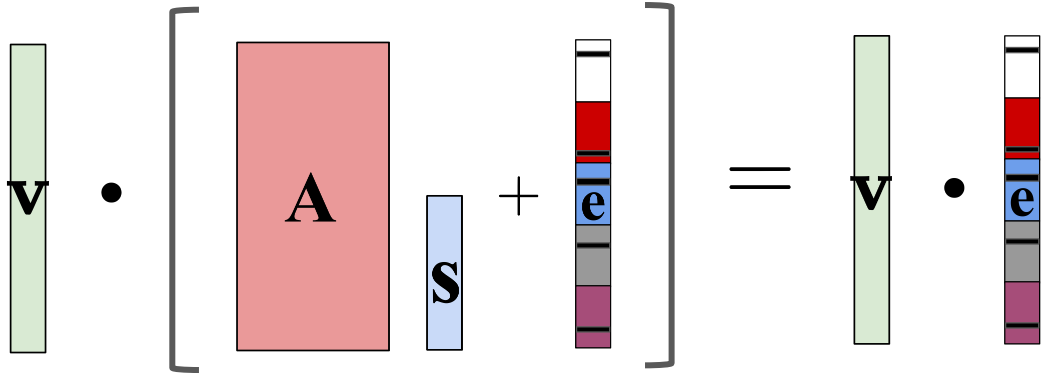


$$\leq \left(1 - \frac{d}{n}\right)^t$$

High-Level Proof Template

$$q = 1$$

codeword **v**

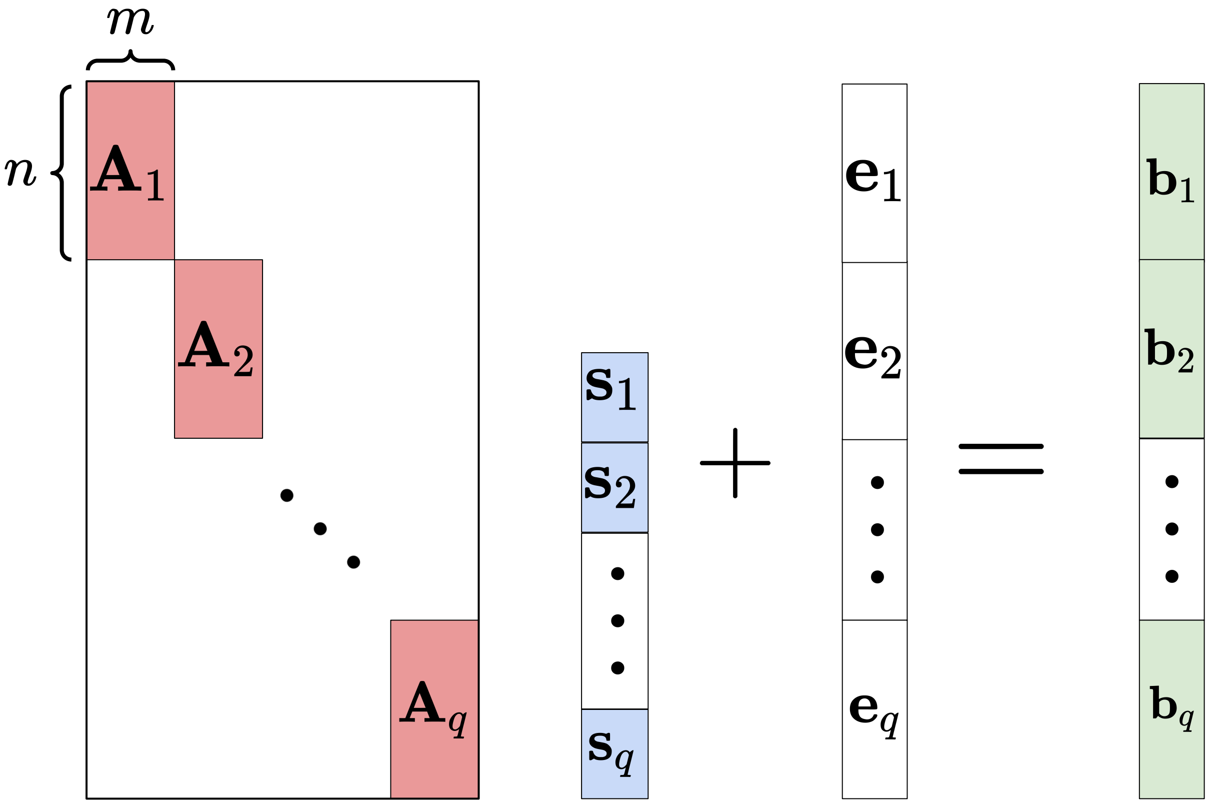


Regular LPN with $\mathbb{F}_2$ noise

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

High-Level Proof Template

Consider canonical representation for q > 1:



High-Level Proof Template

$$q > 1$$

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

$$q > 1$$

v is a concatenation of q codewords

High-Level Proof Template

$$q > 1$$

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

$$q > 1$$

v is a concatenation of q codewords

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

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

Other Linear Attacks

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

Algebraic Attacks

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

Algebraic Attacks

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

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

Algebraic Attacks

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

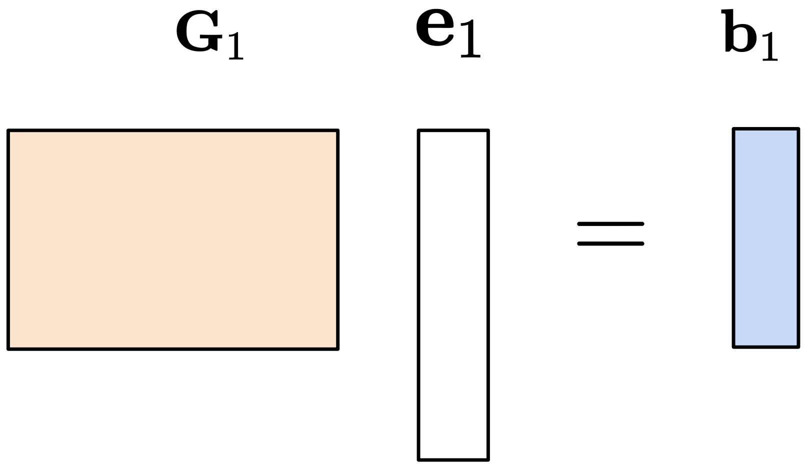
Adapted [BØ23]'s attack to use SSD's additional structure

Bounds on the running time of XL algorithm

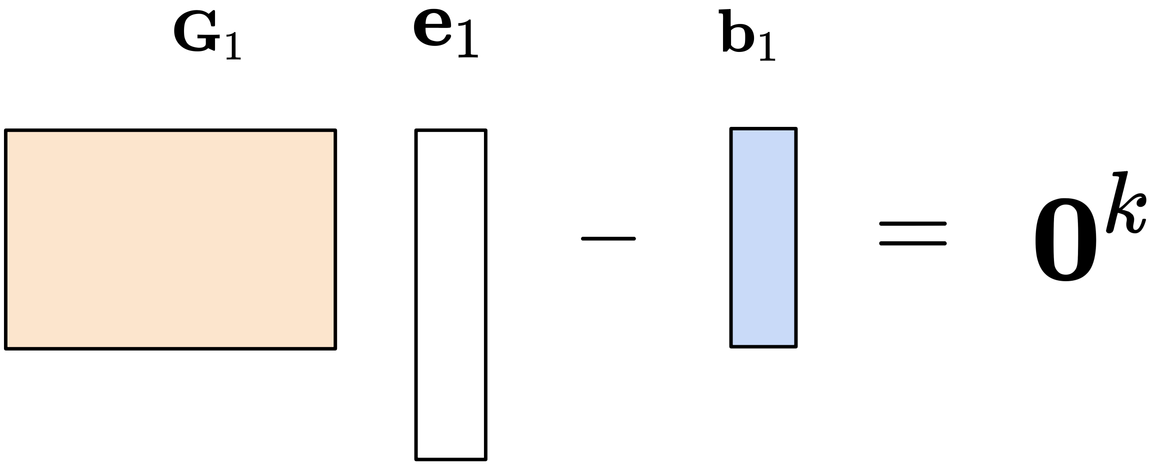
We do not find $q > 1$ reduces security (for PCG parameters)

Not competitive with linear attacks

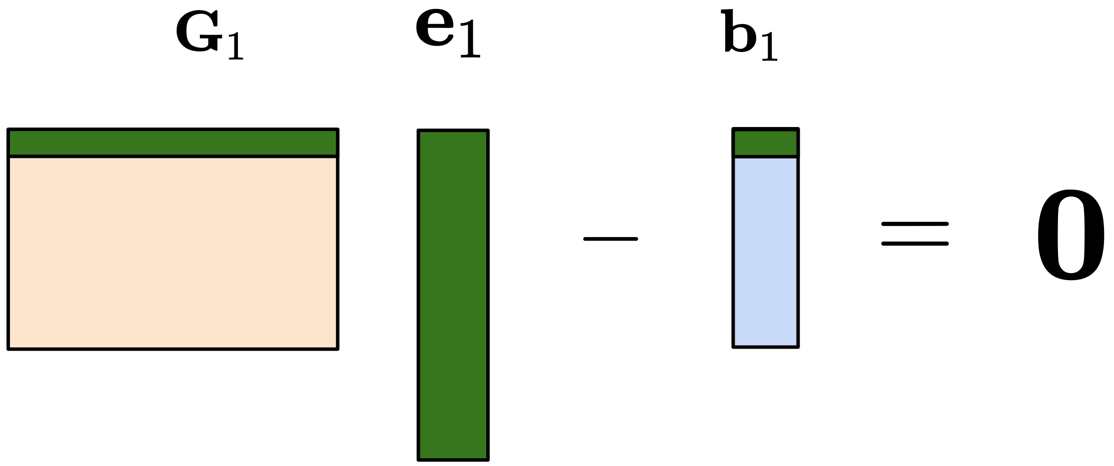
Polynomial System



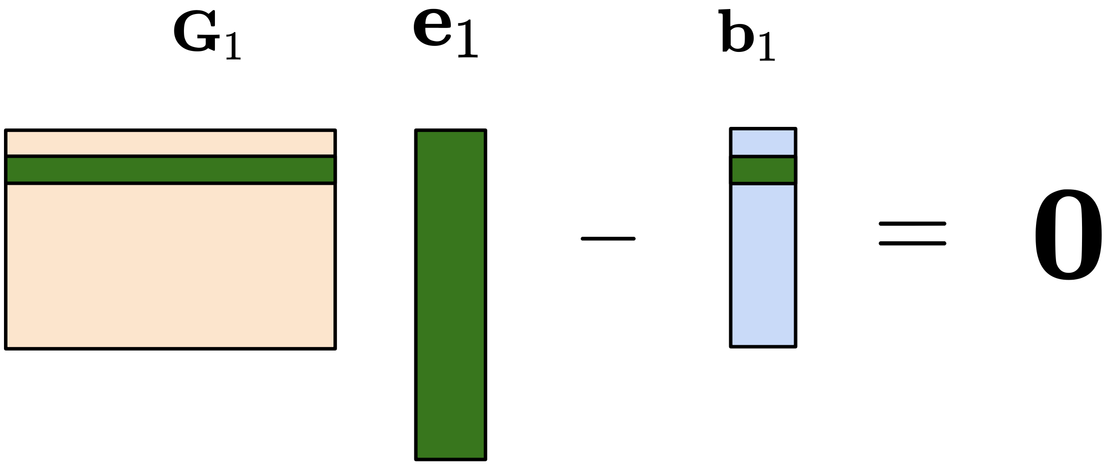
Polynomial System



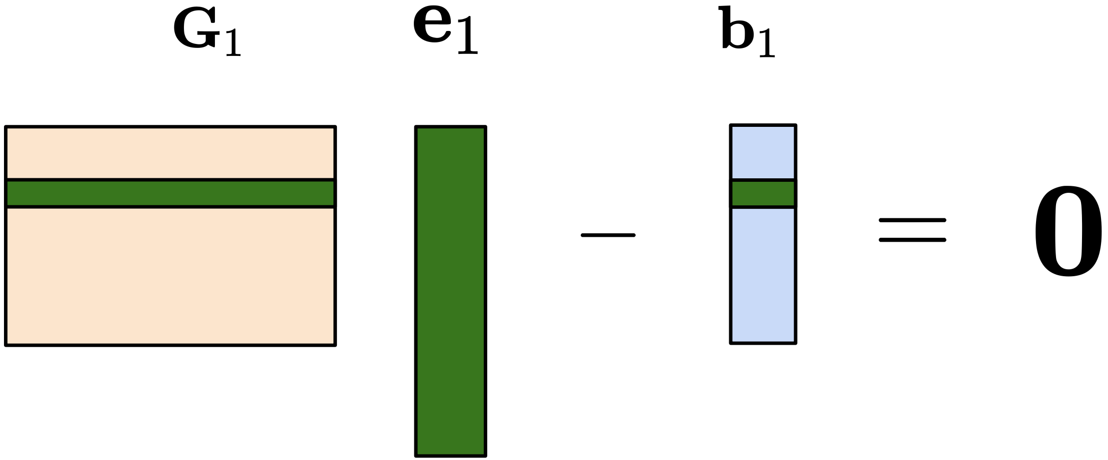
Polynomial System



Polynomial System

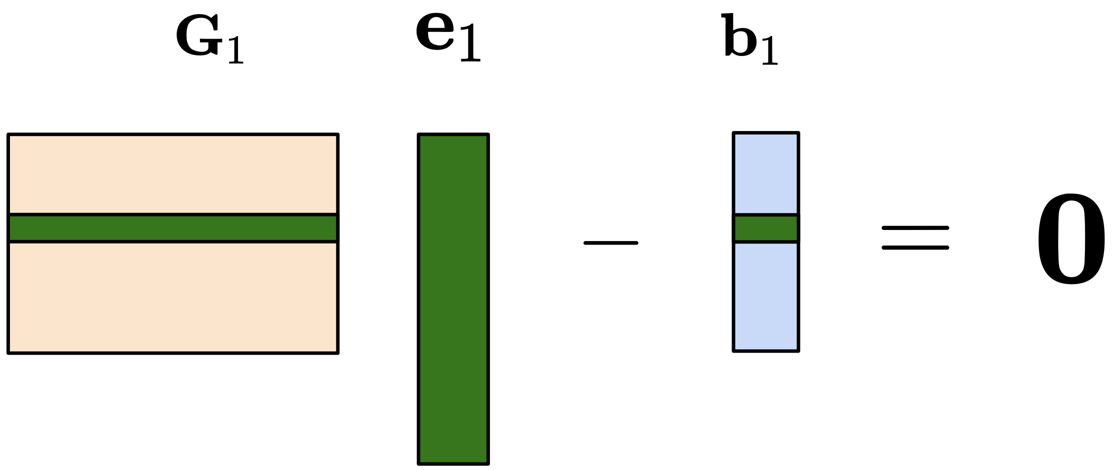


Polynomial System

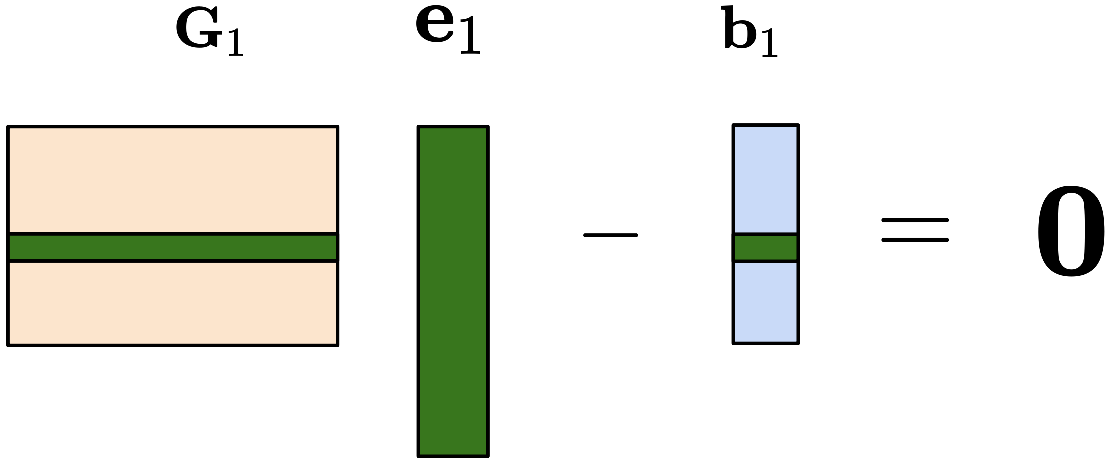


$$\mathbf{G}_1 \mathbf{e}_1 - \mathbf{b}_1 = \mathbf{0}$$

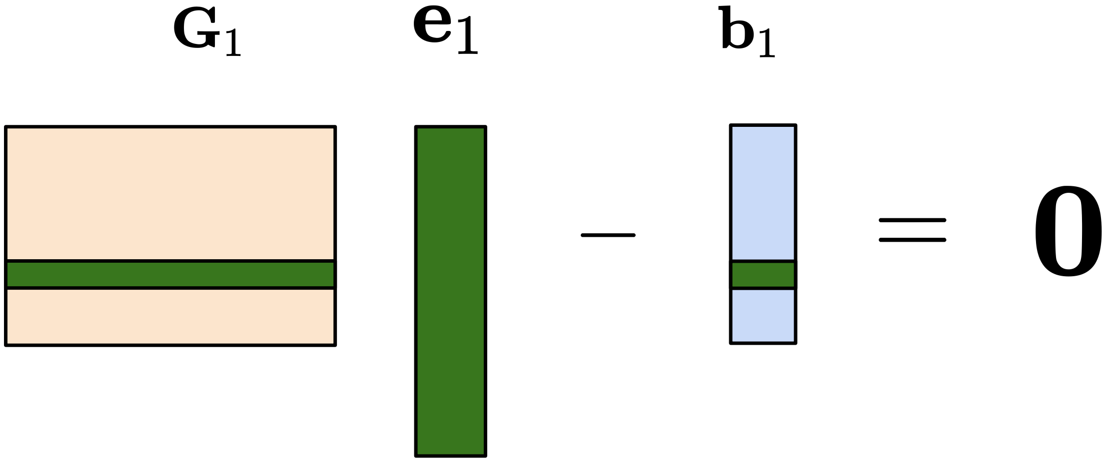
Polynomial System



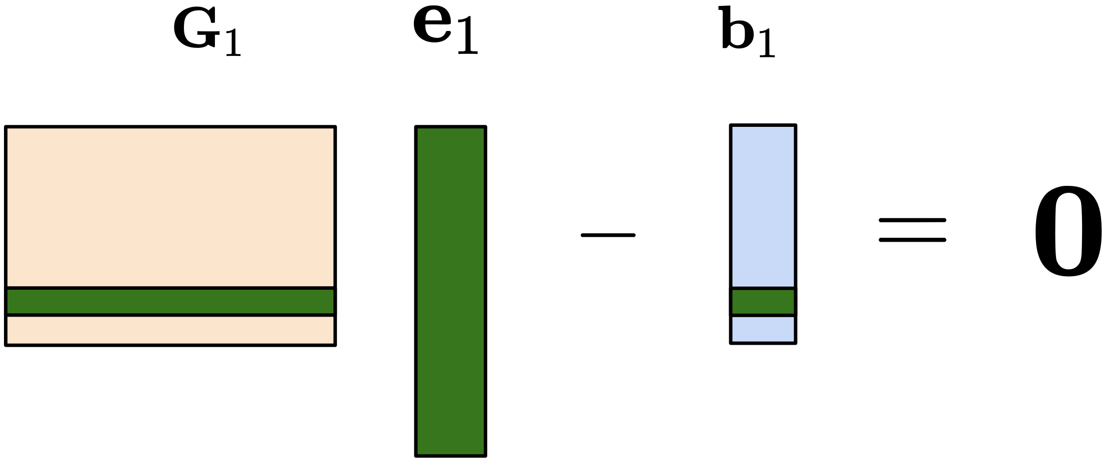
Polynomial System



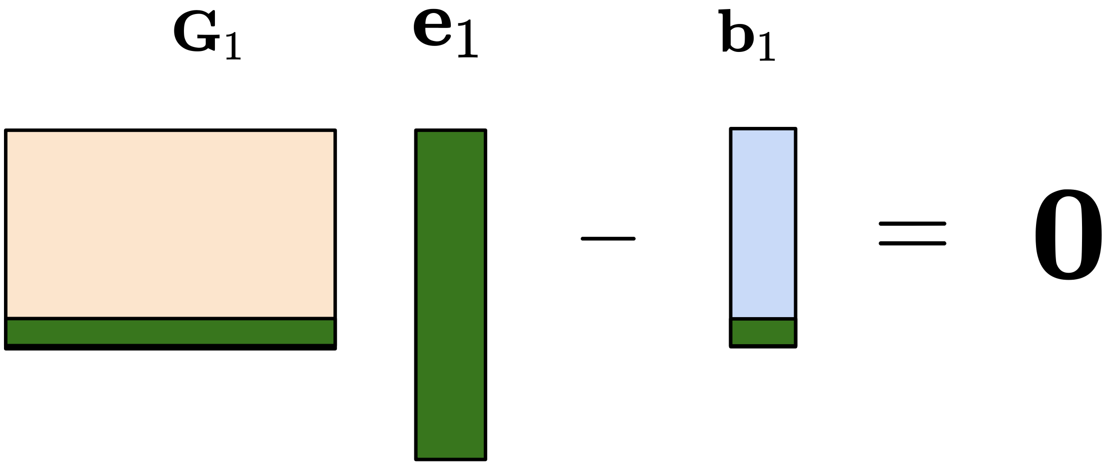
Polynomial System



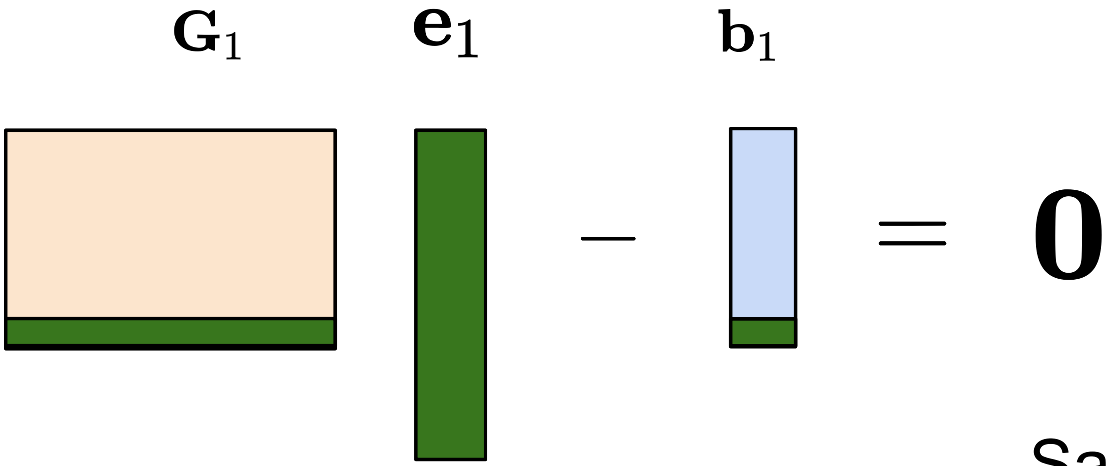
Polynomial System



Polynomial System

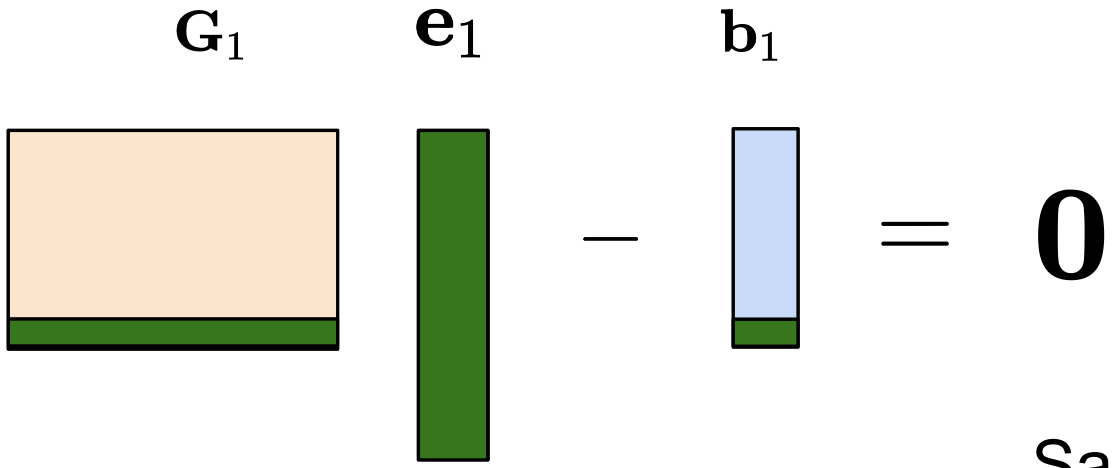


Polynomial System



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

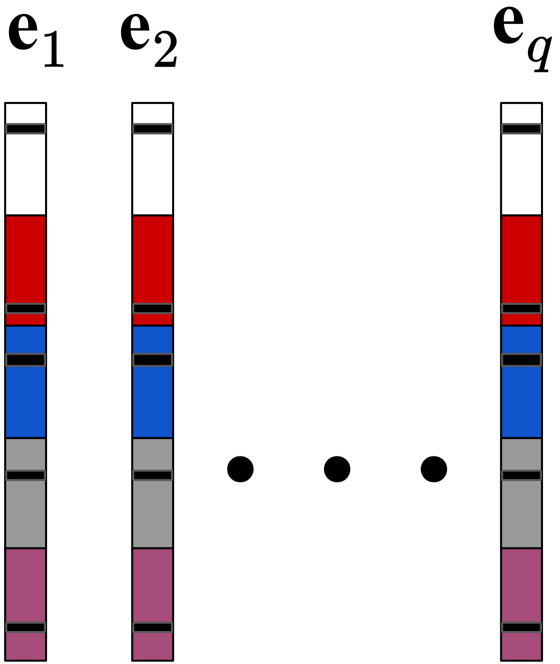
Polynomial System



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

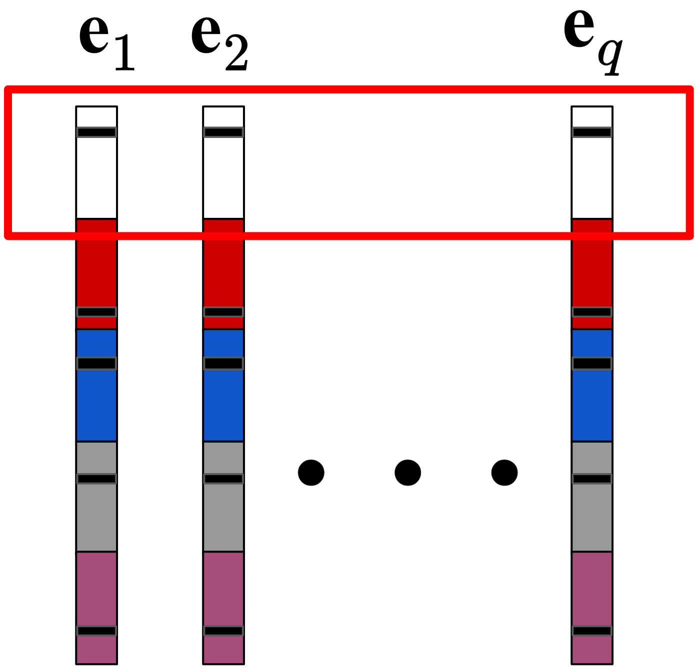
Linear

Polynomial System



Encode stationary structure

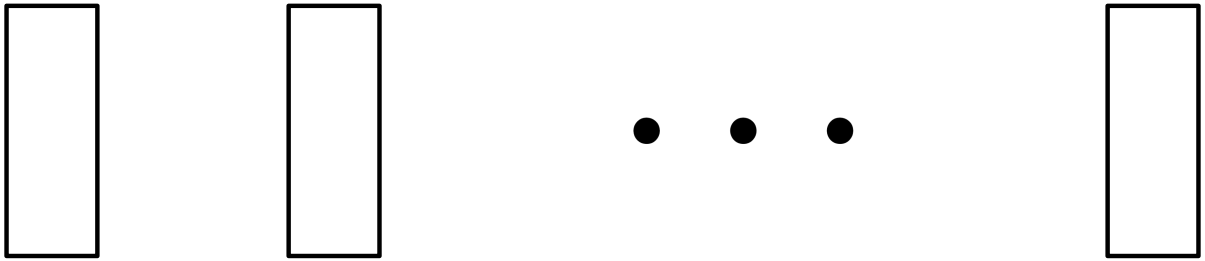
Polynomial System



Encode stationary structure

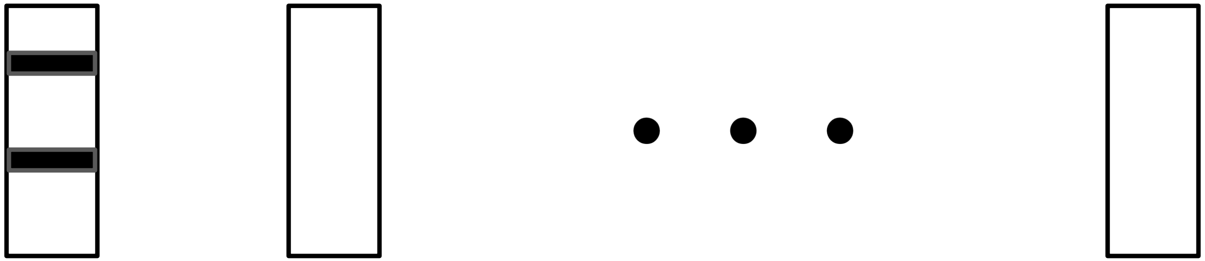
Polynomial System

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



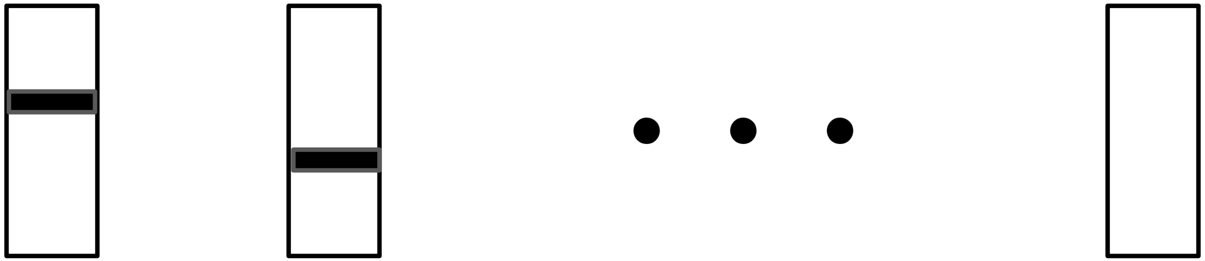
Polynomial System

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



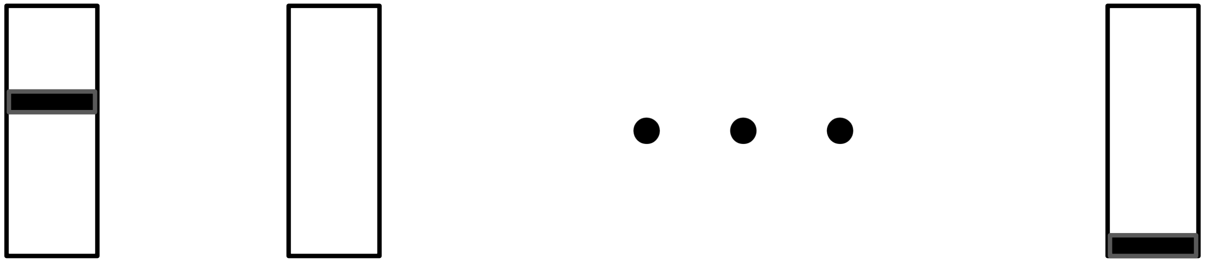
Polynomial System

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



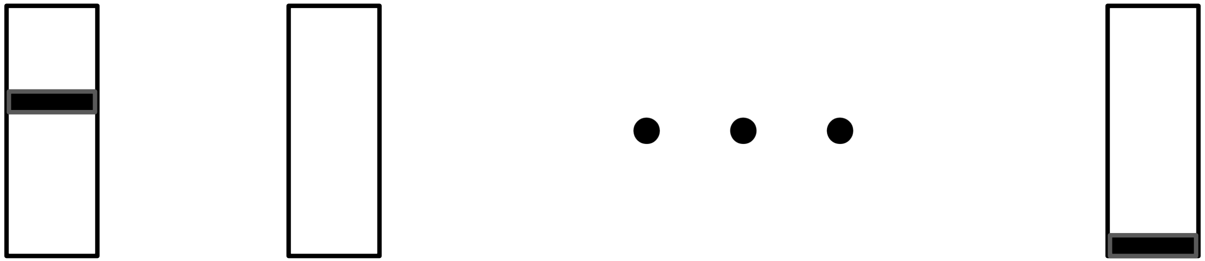
Polynomial System

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



Polynomial System

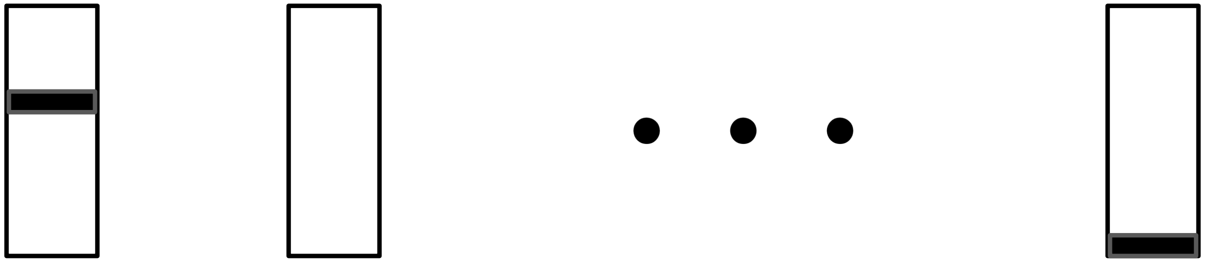
Can we multiply 2 elements in the blocks such that their output is 0?



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 $F=\{f_1, \ldots, f_p\}$

Apply the XL Algorithm [CKPS00]

High-Level Approach

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

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 $F=\{f_1, \ldots, f_p\}$

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 $F=\{f_1, \ldots, f_p\}$

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 $F=\{f_1, \ldots, f_p\}$

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 ≤ 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 $[\Delta \mathbf{e}]$ becomes even more significant

Work in Submission

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

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

Another work closely relies on SSD to generate Beaver triples

Stationary Syndrome Decoding (SSD)

Allows reusing noisy coordinates of **e** across q SD instances

Significant impact on PCG Performance

Stationary Syndrome Decoding (SSD)

Allows reusing noisy coordinates of **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