New Techniques for Obfuscating Conjunctions

James Bartusek (Princeton)

Tancrède Lepoint (SRI & Google)

Fermi Ma (Princeton)

Mark Zhandry (Princeton)

Motivating Scenario: Password Check Program

$\mathrm{P}(x)$:

if $x =$ "correcthorsebatterystaple":

    output 1 (accept)

else:

    output 0 (reject)

Slightly Better Solution

Compute SHA256("correcthorsebatterystaple")

= cbe6beb26479b568e5f15
b50217c6c83c0ee051dc4e5
22b9840d8e291d6aaf46

$\mathrm{P}'(x)$:

if SHA256(x) == "cbe6beb26479b568e5f15b
50217c6c83c0ee051dc4e5
22b9840d8e291d6aaf46":

output 1 (accept)

else:

output 0 (reject)

This is a simple example of ***program obfuscation*** [BGIRSVY] for point functions [Can97,CMR98,LPS04,Wee05,BP12,...]

Informally, want Obf to satisfy:

- (correctness) Obf(P)(x) = P(x) for all x
- (virtual black box) Obf(P) reveals nothing beyond what can be learned from black box access to P

P(x):

if x = "correcthorsebatterystaple"
    output 1 (accept)
else:
    output 0 (reject)

Apply Obf

Obf(P)(x):

if SHA256(x) == "cbe6beb26479b568e5f15b50217c6c83c0ee051dc4e522b9840d8e291d6aaf46":
    output 1 (accept)
else:
    output 0 (reject)

Obfuscation for General Programs

Many candidates:
[GGHRSW13,AGIS14,AB15,Zim15,LV16,Lin16,GMMSSZ16,AS16,LT17,FRS17,BGMZ18,CVW18,AJS18,LM18,Agr18,GJK18,…]

Focus of this work: simple techniques to obfuscate **specific functionalities**.

Obfuscation for Specific Functionalities

- Point Functions
  [Can97,CMR98,LPS04,Wee05,CD08,DKL09,GKPV10,BP12,…]
- Compute-and-Compare Programs
  [GKW17,WZ17]
- Hamming Balls [DS05]
- Hyperplane Membership [CRV09]
- Conjunctions [BVWW16,GKW17,WZ17,…]

We study **conjunctions**, but techniques apply to hamming balls, affine spaces, etc.

Obfuscation for Conjunctions

("pattern-matching with wildcards")

pat = 1*10*

(match) 11100

(mismatch) 10001

(match) 10101

(mismatch) 01111

bitstring x **matches** pat if it equals pat except on *

$P_{pat}(x)$:

if x matches pat output 1

else output 0

Obfuscation for Conjunctions
("pattern-matching with wildcards")

pat = 1*10*
(match) 11100
(mismatch) 10001
(match) 10101
(mismatch) 01111

Our work: Allow evaluation of P_{pat} without leaking anything about pat

bitstring x **matches** pat if it equals pat except on *

$P_{pat}(x)$:
if x matches pat output 1
else output 0

Obfuscation for Conjunctions

("pattern-matching with wildcards")

pat = 1*10*

(match) 11100

(mismatch) 10001

(match) 10101

(mismatch) 01111

bitstring x **matches** pat if it equals pat except on *

$P_{pat}(x)$:

if x matches pat output 1

else output 0

Our work: Allow evaluation of P_{pat} without leaking anything about pat

When is this goal feasible?

A: pat must be drawn from a distribution where accepting inputs to P_{pat} are hard to find [BBCKPS13]

Prior Conjunction Obfuscators

	Assumption or Model
[BR13]	Multilinear Maps
[BVWW16]	Entropic Ring LWE
[GKW17],[WZ17]	LWE
[BKMPRS18]	**Generic Group Model**

Our starting point: the [BKMPRS18] construction

Our Results: Three Constructions

	Assumption or Model	Security holds when pattern is sampled from:
[BKMPRS18]	Generic Group Model	$U_n[cn]$, where $c < 0.774$
Construction 1	Generic Group Model*	$U_n[n - \omega(\log n)]$
Construction 2	Learning Parity with Noise	$U_n[cn]$ where $c < 1$
Construction 3 (see paper)	Information theoretic*	$U_n[n^{\epsilon}]$ where $0 \leq \epsilon < 1$

$U_n[w]$ denotes uniform dist over length n patterns with w wildcards

*can be extended beyond uniform distributions (see also [BeuWee19])

Talk Outline

1. Encoding Conjunctions as Inner Products
2. A Group-Based Construction
3. Security from LPN/RLC

What Does "Simple" Mean?

Obfuscation: On input $pat \in \{0,1,*\}^n$
Output vector $\boldsymbol{v}_{pat}$ over $\mathbb{F}_p$

What Does "Simple" Mean?

Obfuscation: On input $pat \in \{0,1,*\}^n$
Output vector $\boldsymbol{v}_{pat}$ over $\mathbb{F}_p$

Evaluation: On input $x \in \{0,1\}^n$
Write down vector $\boldsymbol{w}_x$
Accept if $\boldsymbol{w}_x^T \boldsymbol{v}_{pat} = 0$

Encoding Conjunctions as Inner Products

Structure of $\boldsymbol{e}$
taken from [BKMPRS18]

“Obfuscation”:

$(n = 3)$

$pat = {*}01$

$$\boldsymbol{e} = \begin{array}{cc} & pat \\ \begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix} & \begin{array}{c} * \\ \\ 0 \\ \\ \\ 1 \end{array} \end{array}$$

r_i is a uniformly random value in $\mathbb{F}_p$

Encoding Conjunctions as Inner Products

Structure of $\boldsymbol{e}$
taken from [BKMPRS18]

"Obfuscation":

$(n = 3)$

$pat = {*}01$

$$\boldsymbol{e} = \begin{array}{cc} & pat \\ \begin{pmatrix} 0 \\ 0 \\ r_1 \\ 0 \\ 0 \\ r_2 \end{pmatrix} & \begin{array}{c} * \\ \\ 0 \\ \\ 1 \\ \end{array} \end{array}$$

r_i is a uniformly random value in $\mathbb{F}_p$

(Accepting input)

Evaluation: $x = 001$

$$\begin{array}{cccccccc} & x = & 0 & & 0 & & 1 & \\ \boldsymbol{w^T} & = & (0 & \$ & 0 & \$ & \$ & 0) \end{array}$$

\$ denotes arbitrary non-zero value in $\mathbb{F}_p$

Encoding Conjunctions as Inner Products

Structure of $\boldsymbol{e}$
taken from [BKMPRS18]

"Obfuscation":

$(n = 3)$

$pat = {*}01$

$$\boldsymbol{e} = \begin{array}{c c} & pat \\ \begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix} & \begin{array}{c} * \\ \\ 0 \\ \\ 1 \\ \\ \end{array} \end{array}$$

r_i is a uniformly random value in $\mathbb{F}_p$

(Accepting input)

Evaluation: $x = 001$

$$\begin{array}{r c} x = & \begin{array}{c c c} 0 & 0 & 1 \end{array} \\ \boldsymbol{w}^{\boldsymbol{T}} \cdot \boldsymbol{e} = & (0 \quad \$ \;\vdots\; 0 \quad \$ \;\vdots\; \$ \quad 0) \end{array} \begin{array}{c c} & pat \\ \begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix} & \begin{array}{c} * \\ \\ 0 \\ \\ 1 \\ \\ \end{array} \end{array}$$

$= 0$ → **Accept!**

$ denotes arbitrary non-zero value in $\mathbb{F}_p$

Encoding Conjunctions as Inner Products

Structure of $\boldsymbol{e}$
taken from [BKMPRS18]

“Obfuscation”:
$(n = 3)$
$pat = {*}01$

$$\boldsymbol{e} = \begin{array}{c} \\ \left(\begin{array}{c} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{array}\right) \end{array} \begin{array}{c} pat \\ * \\ \\ 0 \\ \\ 1 \\ \end{array}$$

r_i is a uniformly random value in $\mathbb{F}_p$

(Rejecting input)

Evaluation: $x = 011$

$$\begin{array}{c} x = \quad 0 \qquad\quad 1 \qquad\quad 1 \\ \boldsymbol{w}^{\boldsymbol{T}} \cdot \boldsymbol{e} = (0 \quad \$ \,\vdots\, \$ \quad 0 \,\vdots\, \$ \quad 0) \end{array} \left(\begin{array}{c} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{array}\right) \begin{array}{c} pat \\ * \\ \\ 0 \\ \\ 1 \end{array}$$

$= \$r_1 \longrightarrow$ **Reject!**

\$ denotes arbitrary non-zero value in $\mathbb{F}_p$

Talk Outline

1. Encoding Conjunctions as Inner Products
2. A Group-Based Construction
3. Security from LPN/RLC

How can we make this construction secure?

Idea: Avoid giving out $\boldsymbol{e}$ in the clear, but still allow user to compute $\boldsymbol{w}^T \cdot \boldsymbol{e}$ for any $\boldsymbol{w}$ that encodes an input x

"Obfuscation":

$(n = 3)$

$pat = *01$

$$\boldsymbol{e} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array}\begin{pmatrix} 0 \\ 0 \\ r_1 \\ 0 \\ 0 \\ r_2 \end{pmatrix} \begin{array}{c} pat \\ * \\ \\ 0 \\ \\ 1 \end{array}$$

Evaluation: $x = 001$

$$\boldsymbol{w}^T \cdot \boldsymbol{e} = \underbrace{\overset{x=\;0}{(0 \quad \$} \;\vdots\; \overset{0}{0 \quad \$} \;\vdots\; \overset{1}{\$ \quad 0)}}_{2n} \begin{pmatrix} 0 \\ 0 \\ r_1 \\ 0 \\ 0 \\ r_2 \end{pmatrix} \begin{array}{c} pat \\ * \\ \\ 0 \\ \\ 1 \end{array}$$

$$= 0$$

Slightly Better Construction

Obfuscation:

1) Encode pat in $\boldsymbol{e}$
2) Give out $\boldsymbol{B} \cdot \boldsymbol{e} \in \mathbb{F}_p^{n+1}$ where $\boldsymbol{B}$ is a public $(n+1)\times(2n)$ matrix satisfying **Property 1**.

$$
\begin{array}{cc}
 & 2n \\
n+1 & \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \\ 1^4 & 2^4 & 3^4 & 4^4 & 5^4 & 6^4 \end{pmatrix}
\end{array}
\begin{array}{cc}
 & pat \\
\begin{pmatrix} 0 \\ 0 \\ \hline r_1 \\ 0 \\ \hline 0 \\ r_2 \end{pmatrix} & \begin{array}{c} * \\ \\ 0 \\ \\ 1 \\ \end{array}
\end{array}
$$

$$\boldsymbol{B} \qquad \boldsymbol{e}$$

Property 1: Any $(n+1)\times(n+1)$ submatrix of $\boldsymbol{B}$ is full rank over $\mathbb{F}_p$ (ex: Vandermonde)

Evaluation:

On input x = 001 pick $\boldsymbol{k} \in \mathbb{F}_p^{n+1}$ so that

$$
\begin{array}{ccccccccc}
 & x = & 0 & & 0 & & & 1 & \\
\boldsymbol{k}^T\boldsymbol{B} = & (& 0 & \$ & 0 & \$ & \$ & 0 &)
\end{array}
\text{ encodes } x
$$

(i.e. solve for $\boldsymbol{k}$ to make these n entries of $\boldsymbol{k}^T\boldsymbol{B}$ equal 0)

Accept if $\boldsymbol{k}^T\boldsymbol{B}\boldsymbol{e} = \boldsymbol{0}$

Why does $\boldsymbol{B}$ help security?

Informal Lemma 1 (No Linear Attacks): If pat is drawn with enough entropy, then for any $\boldsymbol{k} \in \mathbb{F}_p^{n+1}$, $\boldsymbol{k}^T \boldsymbol{B} \boldsymbol{e}$ is a uniformly random scalar.

$$\underbrace{\overset{n+1}{(k_1\ k_2\ k_3\ k_4)}}_{\boldsymbol{k}^T}\ \underbrace{\overset{2n}{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \\ 1^4 & 2^4 & 3^4 & 4^4 & 5^4 & 6^4 \end{pmatrix}}}_{\boldsymbol{B}}\ \underbrace{\begin{pmatrix} 0 \\ 0 \\ r_1 \\ 0 \\ 0 \\ r_2 \end{pmatrix}}_{\boldsymbol{e}} \quad \begin{array}{c} pat \\ * \\ 0 \\ 1 \end{array}$$

Property 1: Any $(n+1)\times(n+1)$ submatrix of $\boldsymbol{B}$ is full rank over $\mathbb{F}_p$ (ex: Vandermonde)

Why does $\boldsymbol{B}$ help security?

Informal Lemma 1 (No Linear Attacks): If pat is drawn with enough entropy, then for any $\boldsymbol{k} \in \mathbb{F}_p^{n+1}$, $\boldsymbol{k}^T \boldsymbol{B} \boldsymbol{e}$ is a uniformly random scalar.

($\boldsymbol{B}_i$ denotes ith column of $\boldsymbol{B}$)

$$\underbrace{\overbrace{\left(\boldsymbol{k}^T\boldsymbol{B}_1 \quad \boldsymbol{k}^T\boldsymbol{B}_2 \quad \boldsymbol{k}^T\boldsymbol{B}_3 \quad \boldsymbol{k}^T\boldsymbol{B}_4 \quad \boldsymbol{k}^T\boldsymbol{B}_5 \quad \boldsymbol{k}^T\boldsymbol{B}_6\right)}^{2n}}_{\boldsymbol{k}^T\boldsymbol{B}} \underbrace{\begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix}}_{\boldsymbol{e}} \quad \begin{array}{c} pat \\ * \\ 0 \\ 1 \end{array}$$

1) At most n out of $2n$ entries of $\boldsymbol{k}^T\boldsymbol{B}$ can be 0 (**Property 1**).
2) If pat has enough entropy, then with overwhelming probability one of the n non-zero entries of $\boldsymbol{k}^T\boldsymbol{B}$ will coincide with a non-zero entries in $\boldsymbol{e}$.
3) If so, $(\boldsymbol{k}^T\boldsymbol{B})\boldsymbol{e}$ will be a random scalar.

Property 1: Any $(n+1)\times(n+1)$ submatrix of $\boldsymbol{B}$ is full rank over $\mathbb{F}_p$ (ex: Vandermonde)

Why does $\boldsymbol{B}$ help security?

Informal Lemma 1 (No Linear Attacks): If pat is drawn with enough entropy, then for any $\boldsymbol{k} \in \mathbb{F}_p^{n+1}$, $\boldsymbol{k}^T \boldsymbol{B}\boldsymbol{e}$ is a uniformly random scalar.

Group-Based* Construction

Obfuscation: Encode pat as $\boldsymbol{e}$, compute $\boldsymbol{B}\boldsymbol{e}$ and output:

$$g^{\boldsymbol{B}\boldsymbol{e}} = g^{(\boldsymbol{B}\boldsymbol{e})_1}, g^{(\boldsymbol{B}\boldsymbol{e})_2}, \ldots, g^{(\boldsymbol{B}\boldsymbol{e})_{n+1}}$$

(same evaluation procedure works in exponent)

Proof: generic adversaries limited to linear attacks

Theorem: Generic Group adversary [Nac94,Sho97] cannot distinguish $g^{\boldsymbol{B}\boldsymbol{e}}$ from $n+1$ random group elements if pat is uniformly random** with $n - \omega(\log n)$ wildcards.

*Idea due to [BKMPRS18]: this construction can be viewed as "dual" to their construction.

**Can be extended to more general distributions (see our paper and [BeuWee19])

Talk Outline

1. Encoding Conjunctions as Inner Products
2. A Group-Based Construction
3. Security from LPN/RLC

Group-Based Construction

Step 1: Sample a length $2n$ vector $\boldsymbol{e}$:

If $pat_i = *$, $\begin{pmatrix} e_{2i-1} \\ e_{2i} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

If $pat_i = 0$, $\begin{pmatrix} e_{2i-1} \\ e_{2i} \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}$ for $r \leftarrow \mathbb{F}_p$

If $pat_i = 1$, $\begin{pmatrix} e_{2i-1} \\ e_{2i} \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$ for $r \leftarrow \mathbb{F}_p$

Step 2: Define $\boldsymbol{B} \in \mathbb{F}_p^{(n+1)\times 2n}$ whose (i,j)th entry is:

$$\boldsymbol{B}_{i,j} = j^i$$

Compute the vector $\boldsymbol{Be} \in \mathbb{F}_p^{n+1}$

$$\overset{\boldsymbol{v}}{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}} = \overset{\boldsymbol{B}}{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \\ 1^4 & 2^4 & 3^4 & 4^4 & 5^4 & 6^4 \end{pmatrix}} \overset{\boldsymbol{e}}{\begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix}} \begin{array}{c} * \\ \\ 0 \\ \\ 1 \\ \\ \end{array}$$

Obfuscation: $g^{\boldsymbol{Be}}$

($\boldsymbol{B}$ is a fixed public matrix)

Group-Based Construction

Step 1: Sample a length $2n$ vector $\boldsymbol{e}$:

If $pat_i = *$, $\begin{pmatrix} e_{2i-1} \\ e_{2i} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

If $pat_i = 0$, $\begin{pmatrix} e_{2i-1} \\ e_{2i} \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}$ for $r \leftarrow \mathbb{F}_p$

If $pat_i = 1$, $\begin{pmatrix} e_{2i-1} \\ e_{2i} \end{pmatrix} = \begin{pmatrix} 0 \\ r \end{pmatrix}$ for $r \leftarrow \mathbb{F}_p$

Step 2: Define $\boldsymbol{B} \in \mathbb{F}_p^{(n+1)\times 2n}$ whose (i, j)th entry is:

$$\boldsymbol{B}_{i,j} = j^i$$

Compute the vector $\boldsymbol{B}\boldsymbol{e} \in \mathbb{F}_p^{n+1}$

$$\overset{\boldsymbol{v}}{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}} = \overset{\boldsymbol{B}}{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1^2 & 2^2 & 3^2 & 4^2 & 5^2 & 6^2 \\ 1^3 & 2^3 & 3^3 & 4^3 & 5^3 & 6^3 \\ 1^4 & 2^4 & 3^4 & 4^4 & 5^4 & 6^4 \end{pmatrix}} \overset{\boldsymbol{e}}{\begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix}} \begin{matrix} * \\ \\ 0 \\ \\ 1 \end{matrix}$$

Obfuscation: $\boldsymbol{B}\boldsymbol{e}$?

($\boldsymbol{B}$ is a fixed public matrix)

New Construction

Step 1: Sample a length $2n$ vector $\boldsymbol{e}$:

If $pat_i = *$, $\binom{e_{2i-1}}{e_{2i}} = \binom{0}{0}$

If $pat_i = 0$, $\binom{e_{2i-1}}{e_{2i}} = \binom{r}{0}$ for $r \leftarrow \mathbb{F}_p$

If $pat_i = 1$, $\binom{e_{2i-1}}{e_{2i}} = \binom{0}{r}$ for $r \leftarrow \mathbb{F}_p$

Step 2: Sample a ***uniformly random*** matrix $\boldsymbol{B} \leftarrow \mathbb{F}_p^{(n+1)\times 2n}$.

Compute the vector $\boldsymbol{B}\boldsymbol{e} \in \mathbb{F}_p^{n+1}$

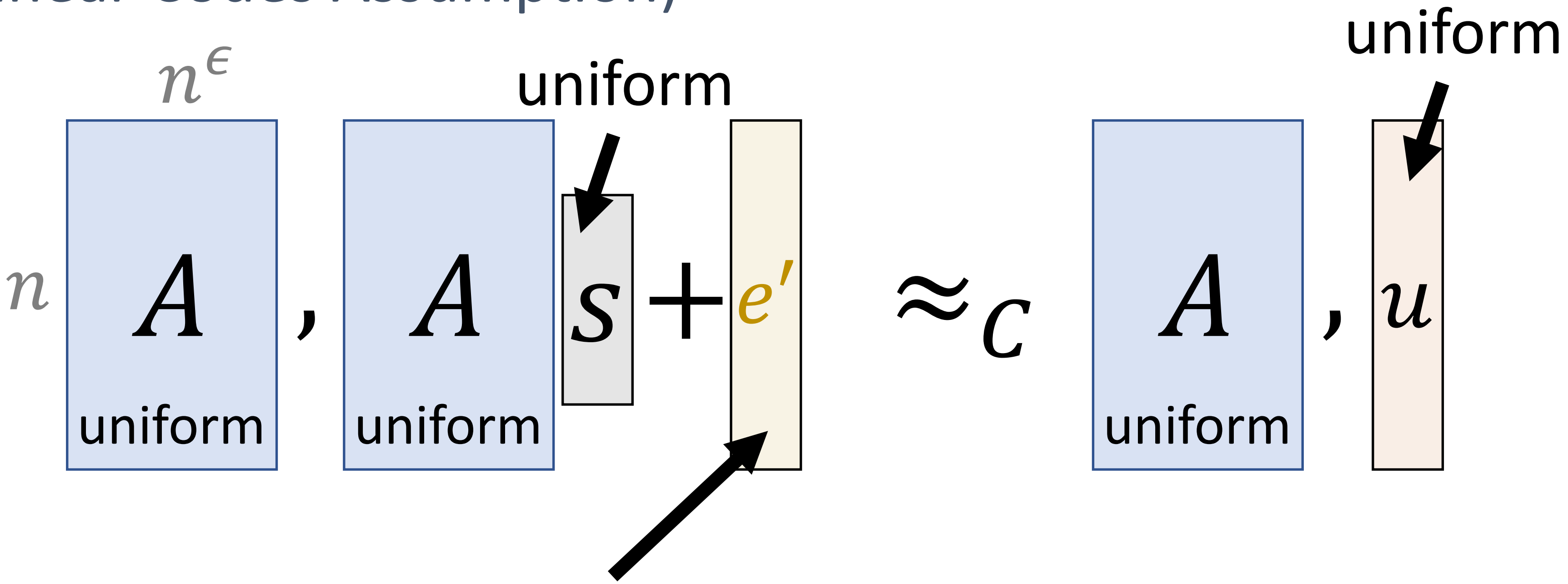
$$\underset{\boldsymbol{v}}{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}} = \underset{\boldsymbol{B}}{\begin{pmatrix} \quad \text{random } \boldsymbol{B} \quad \end{pmatrix}} \underset{\boldsymbol{e}}{\begin{pmatrix} 0 \\ 0 \\ r_1 \\ 0 \\ 0 \\ r_2 \end{pmatrix}} \begin{matrix} * \\ \\ 0 \\ \\ 1 \\ \end{matrix}$$

Obfuscation: $\boldsymbol{B}, \boldsymbol{B}\boldsymbol{e}$

Idea: Randomize $\boldsymbol{B}$!

Why would this be secure?

Learning Parity with Noise Assumption over $\mathbb{F}_p$
(Random Linear Codes Assumption)

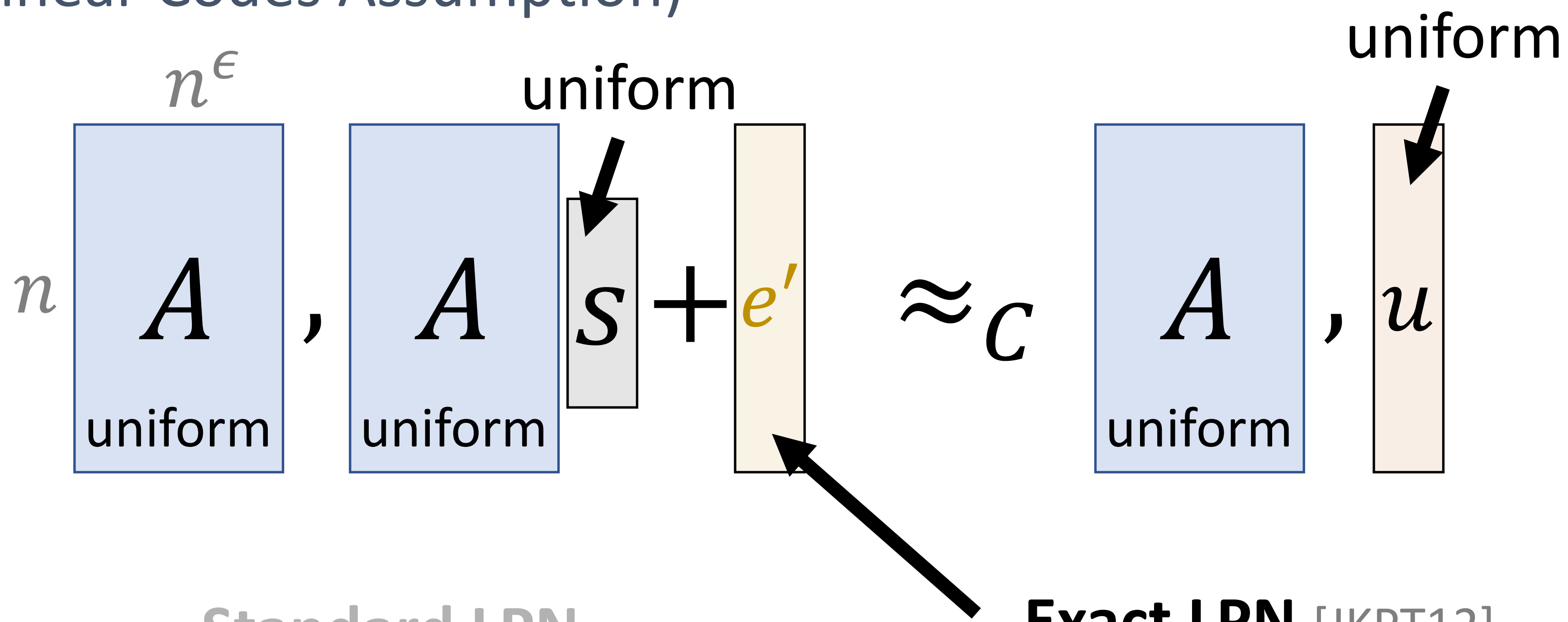


Standard LPN

set each entry

- 0 w/ prob $1-\alpha$
- $e'_i \leftarrow \mathbb{F}_p$ w/ prob α

Learning Parity with Noise Assumption over $\mathbb{F}_p$ (Random Linear Codes Assumption)



Standard LPN

set each entry

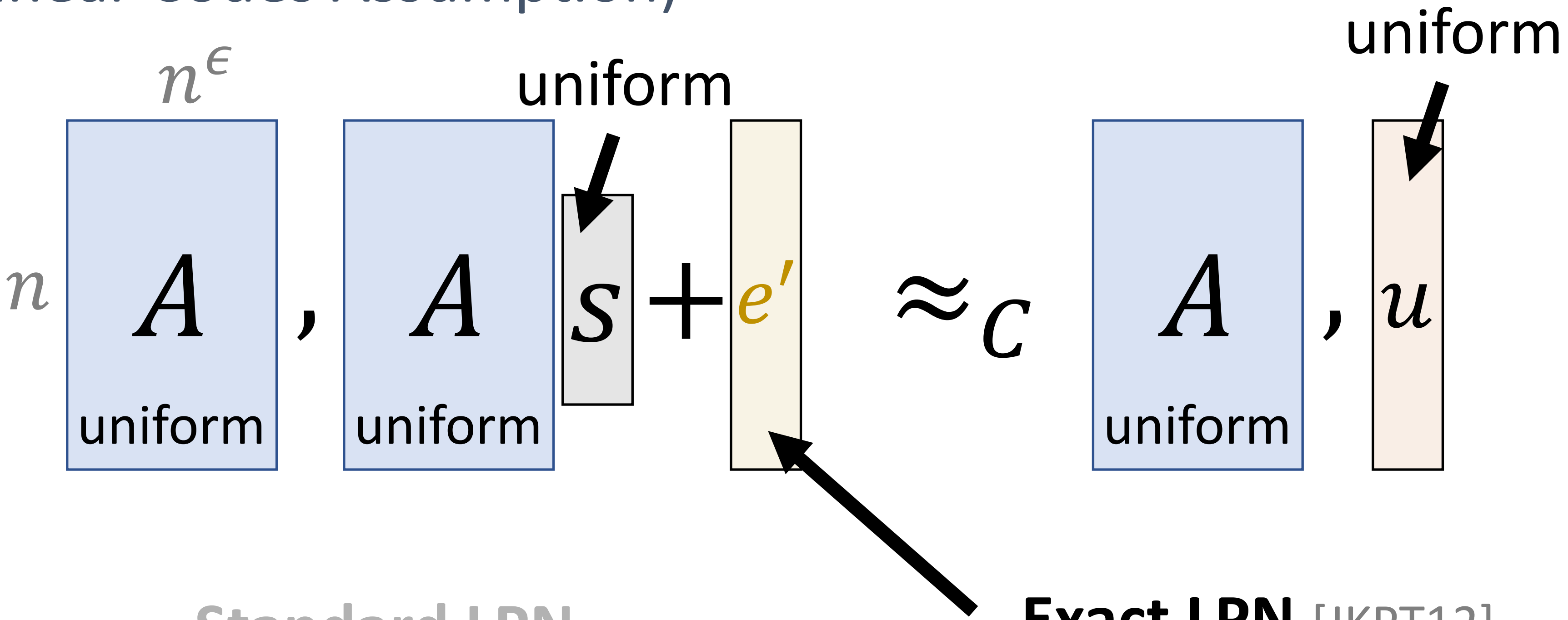
- 0 w/ prob $1 - \alpha$
- $e'_i \leftarrow \mathbb{F}_p$ w/ prob α

Exact LPN [JKPT12]

set exactly αn entries non-zero

(polynomially equivalent)

Learning Parity with Noise Assumption over $\mathbb{F}_p$ (Random Linear Codes Assumption)



Standard LPN

set each entry

- 0 w/ prob $1-\alpha$
- $e'_i \leftarrow \mathbb{F}_p$ w/ prob α

Exact LPN [JKPT12]

set exactly αn entries non-zero

(polynomially equivalent)

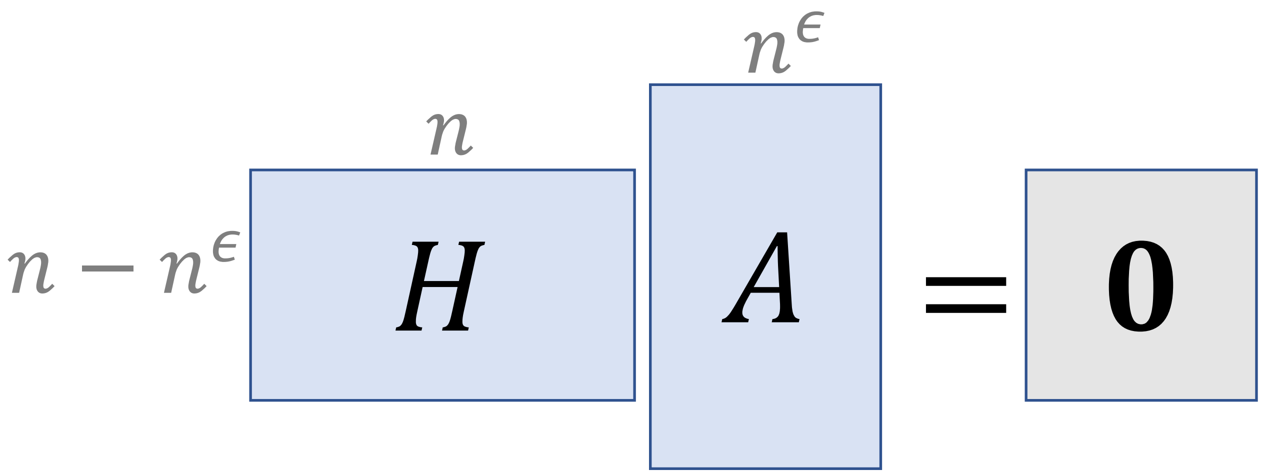
Last modification: switch to “dual”

Compute $\boldsymbol{H}$ with full row-rank such that:

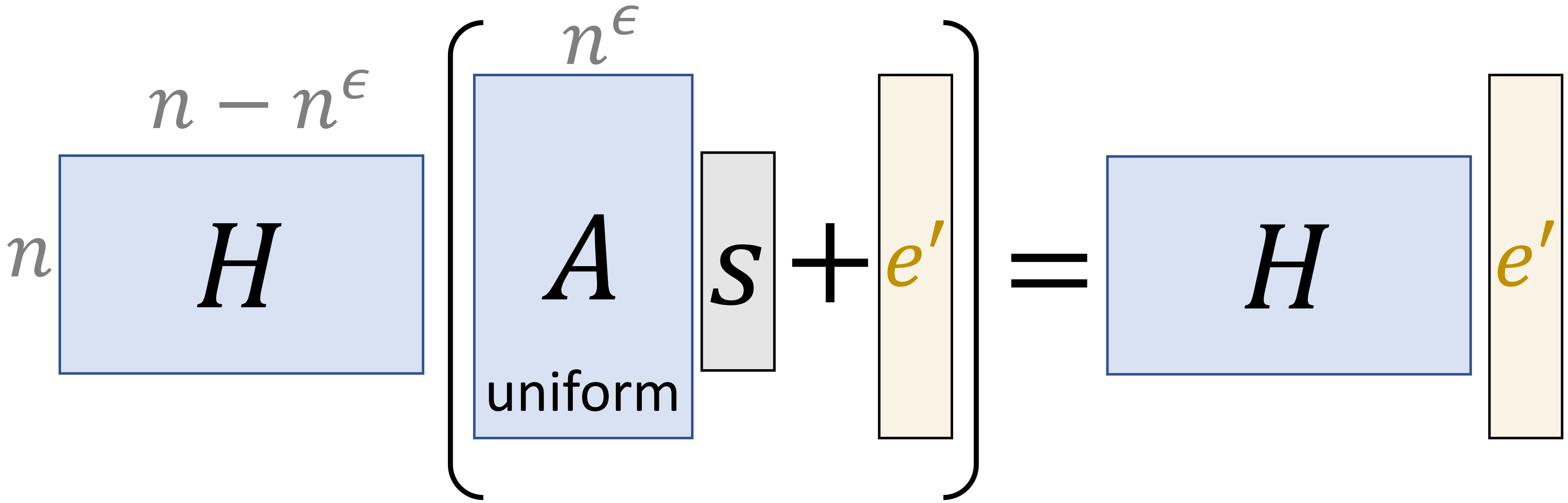
$$\begin{array}{cc} & n \\ n - n^{\epsilon} & \boxed{\;H\;} \end{array} \begin{array}{c} n^{\epsilon} \\ \boxed{A} \end{array} = \boxed{\mathbf{0}}$$

Last modification: switch to “dual”

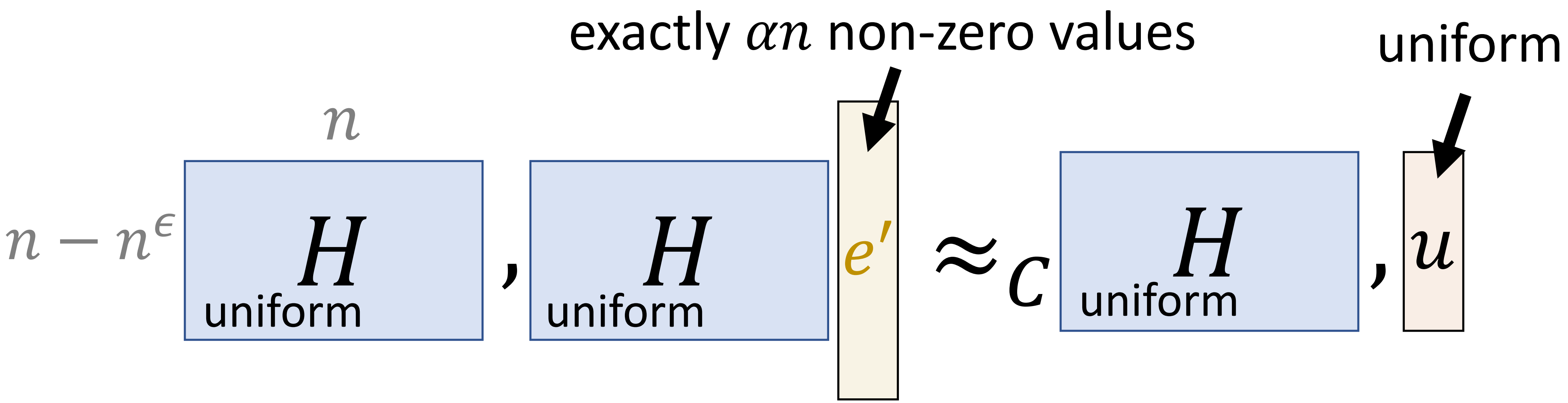
Compute $\boldsymbol{H}$ with full row-rank such that:



Observe



Last modification: switch to "dual"



(Dual) Exact LPN Assumption

(polynomially equivalent to LPN)

Notice $\boldsymbol{H}, \boldsymbol{H}\boldsymbol{e}'$ looks like the obfuscation $\boldsymbol{B}, \boldsymbol{B}\boldsymbol{e}$

(Dual) Exact LPN

$$\underset{\boldsymbol{v}}{\begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_{n-n^\epsilon} \end{pmatrix}} = \underset{\boldsymbol{H}}{\begin{pmatrix} \text{random } \boldsymbol{H} \end{pmatrix}} \underset{\boldsymbol{e'}}{\begin{pmatrix} r_1 \\ 0 \\ r_2 \\ r_3 \\ 0 \end{pmatrix}}$$

LPN for poly samples, field $\mathbb{F}_p$
constant noise α, [JKPT12] "exact" error

$\boldsymbol{H}$ is $\boxed{(n - n^\epsilon) \times n}$

- Sample random $\boldsymbol{H}$ over $\mathbb{F}_p$
- Sample $\boldsymbol{e'}$ as uniformly random $\boxed{n}$ dimensional vector with exactly αn non-zero entries.

Dual Exact LPN Assumption:
$(\boldsymbol{H}, \boldsymbol{H}\boldsymbol{e'})$ looks random

Obfuscation

$$\underset{\boldsymbol{v}}{\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}} = \underset{\boldsymbol{B}}{\begin{pmatrix} \text{random } \boldsymbol{B} \end{pmatrix}} \begin{array}{c} \boldsymbol{e} \\ \begin{pmatrix} 0 \\ 0 \\ \hdashline r_1 \\ 0 \\ \hdashline 0 \\ r_2 \end{pmatrix} \end{array} \begin{array}{c} * \\ \\ 0 \\ \\ 1 \end{array}$$

random length n pattern,
$(1-\alpha)n$ wildcards

$\boldsymbol{B}$ is $\boxed{(n+1) \times 2n}$

- Sample random $\boldsymbol{B}$ over $\mathbb{F}_p$
- Sample $\boldsymbol{e}$ as uniformly random $\boxed{2n}$ dimensional vector with exactly αn non-zero entries, conditioned on each pair of positions $2\mathrm{i}-1, 2\mathrm{i}$ having at least one 0 entry.

(Dual) Exact LPN

"unstructured error"

uniformly random n-dimensional vector with exactly αn non-zero entries.

$$\boldsymbol{e'} = \begin{pmatrix} e'_1 \\ 0 \\ e'_2 \\ e'_3 \\ 0 \end{pmatrix}$$

Obfuscation

"structured error"

uniformly random $2n$-dimensional vector with exactly αn non-zero entries conditioned on each pair of positions $2\mathrm{i}-1, 2\mathrm{i}$ having at least one 0 entry.

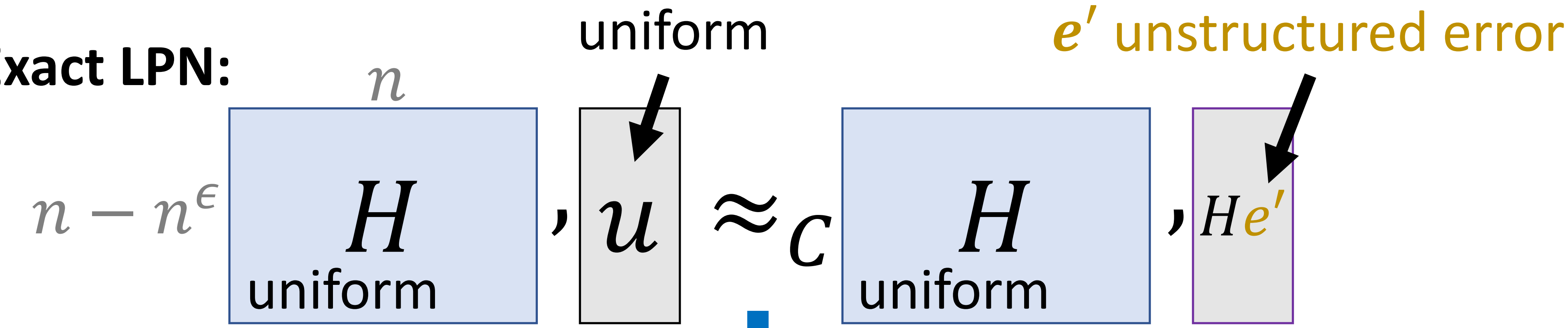
$$\boldsymbol{e} = \begin{pmatrix} e_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ e_2 \\ e_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

This distribution arises if pat is uniformly random with $(1-\alpha)n$ wildcards.

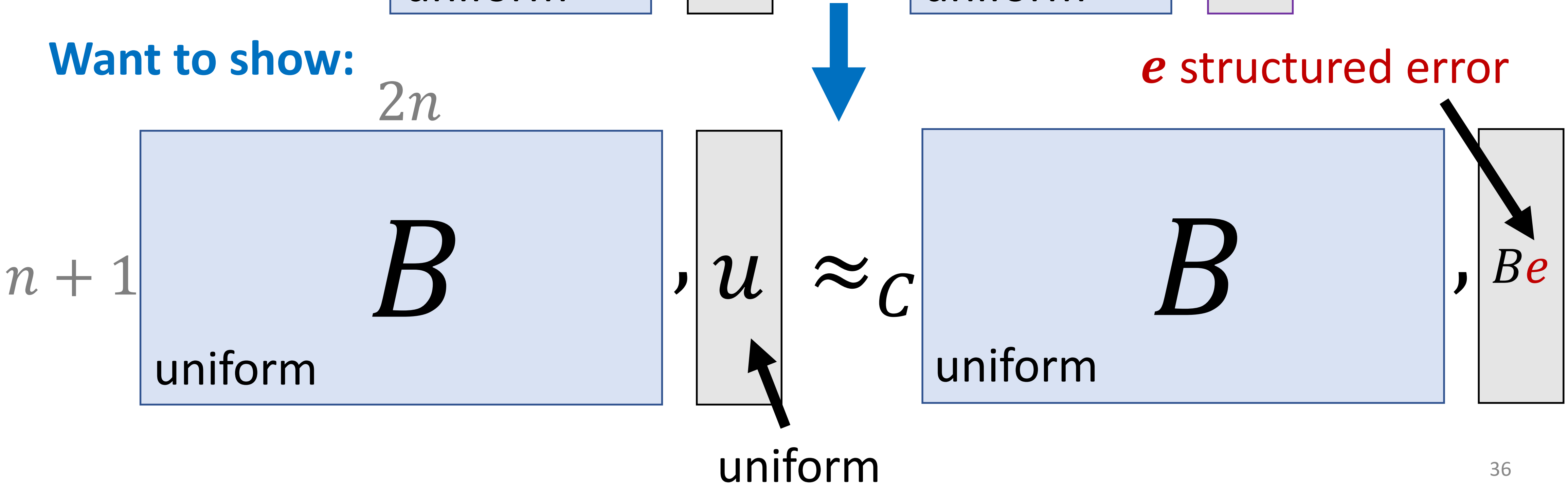
Theorem: Assuming LPN over $\mathbb{F}_p$ (noise rate α), obfuscation $\boldsymbol{B}, \boldsymbol{B}\boldsymbol{e}$ looks uniformly random if pat is uniformly at random with $(1-\alpha)n$ wildcards, for $0 < \alpha < 1$.

Theorem: Assuming LPN over $\mathbb{F}_p$ (noise rate α), obfuscation $\boldsymbol{B}, \boldsymbol{Be}$ looks uniformly random if pat is uniformly at random with $(1-\alpha)n$ wildcards, for $0 < \alpha < 1$.

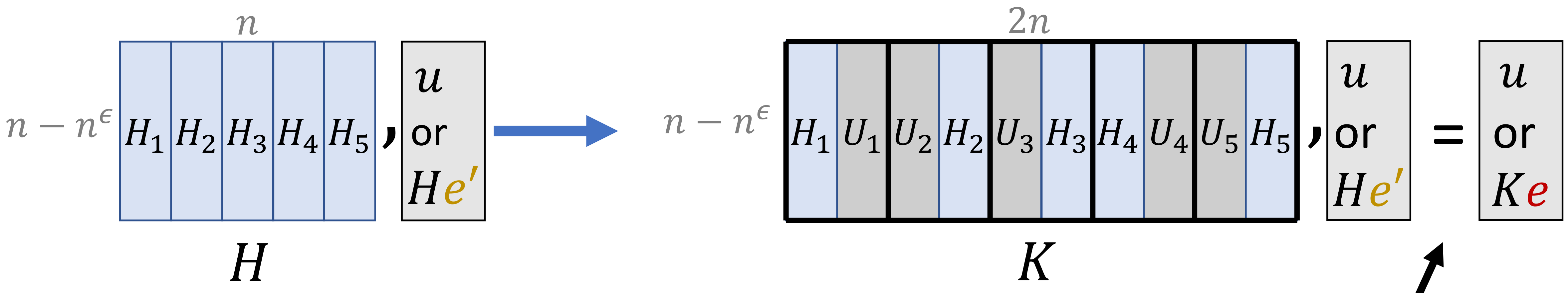
(Dual) Exact LPN:



Want to show:



Easy Step: Sample n random columns $U_1, \ldots U_n$. Replace $\boldsymbol{H}$ with $\boldsymbol{K}$ where each **pair of indices** $(2i-1, 2i)$ is either H_i, U_i or U_i, H_i (pick randomly).



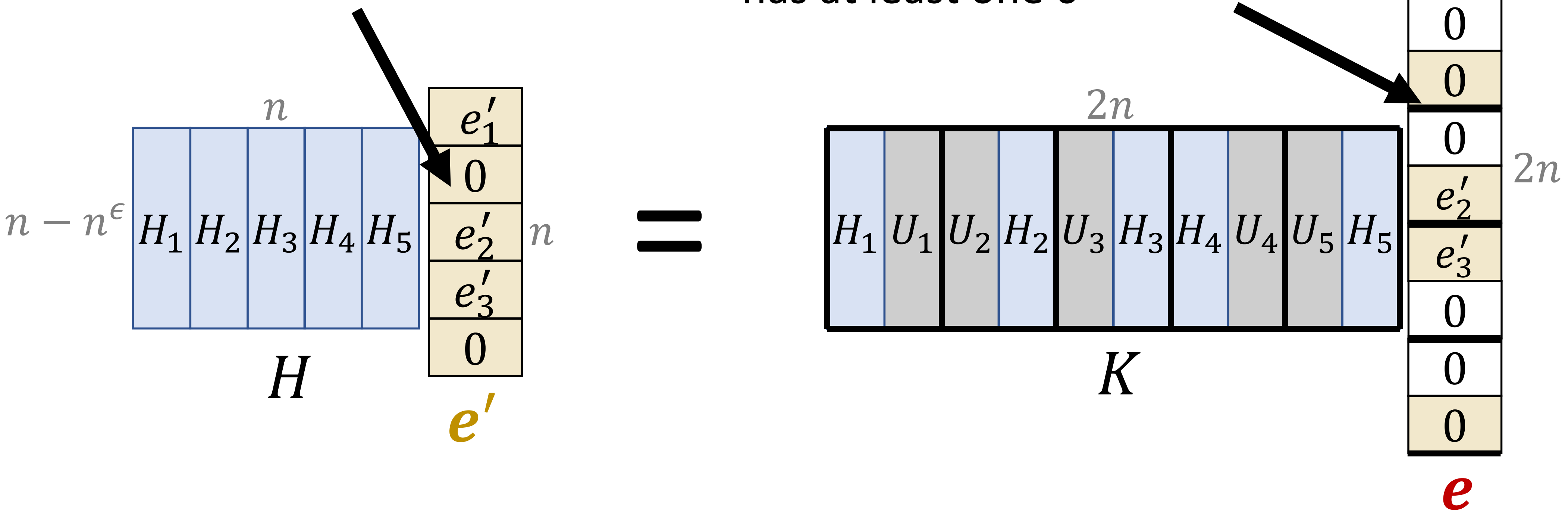
Claim: $\boldsymbol{H}\boldsymbol{e}' = \boldsymbol{K}\boldsymbol{e}$ where $\boldsymbol{e}'$ is unstructured error and $\boldsymbol{e}$ is structured error.

e **unstructured error**

non-zero entries in αn randomly chosen positions

e **structured error**

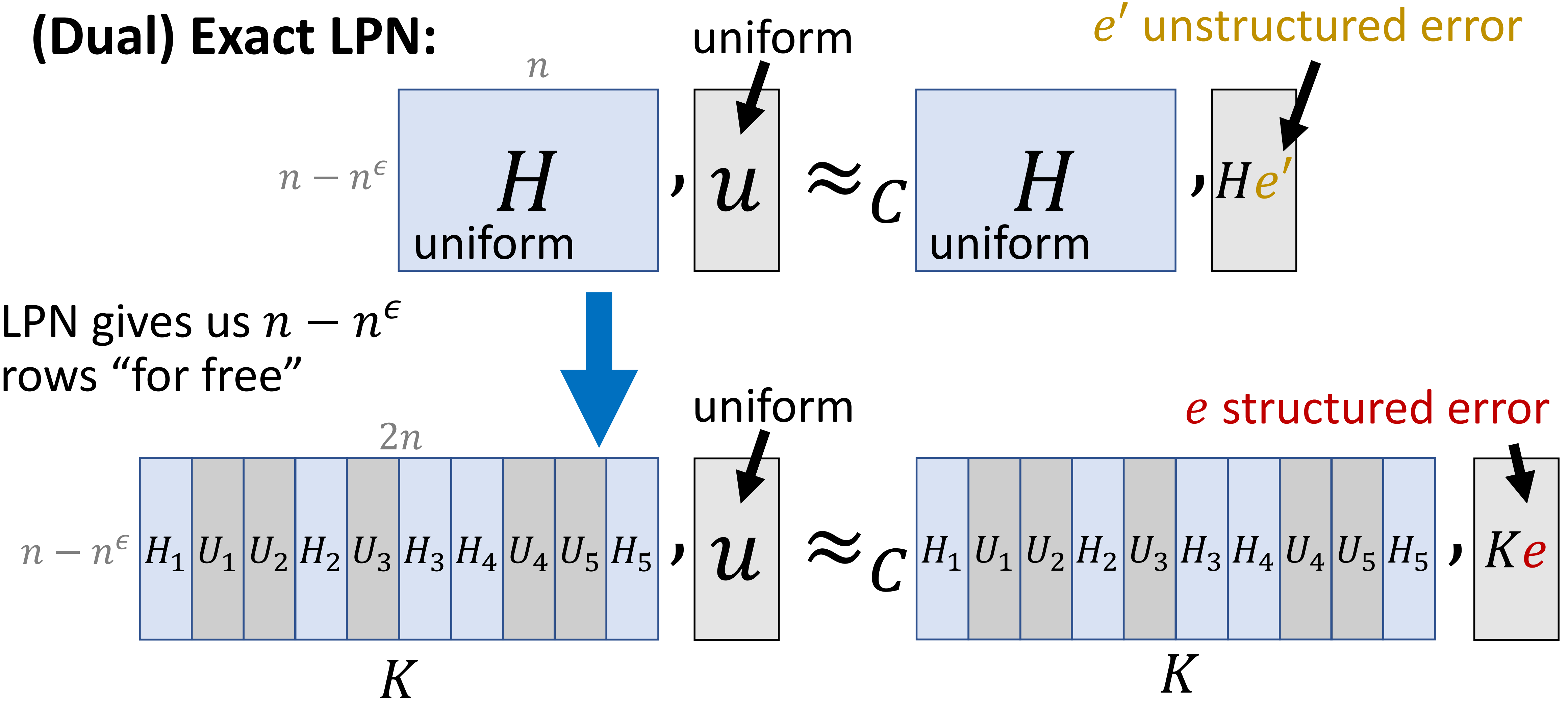
non-zero entries in αn randomly chosen positions, each pair has at least one 0



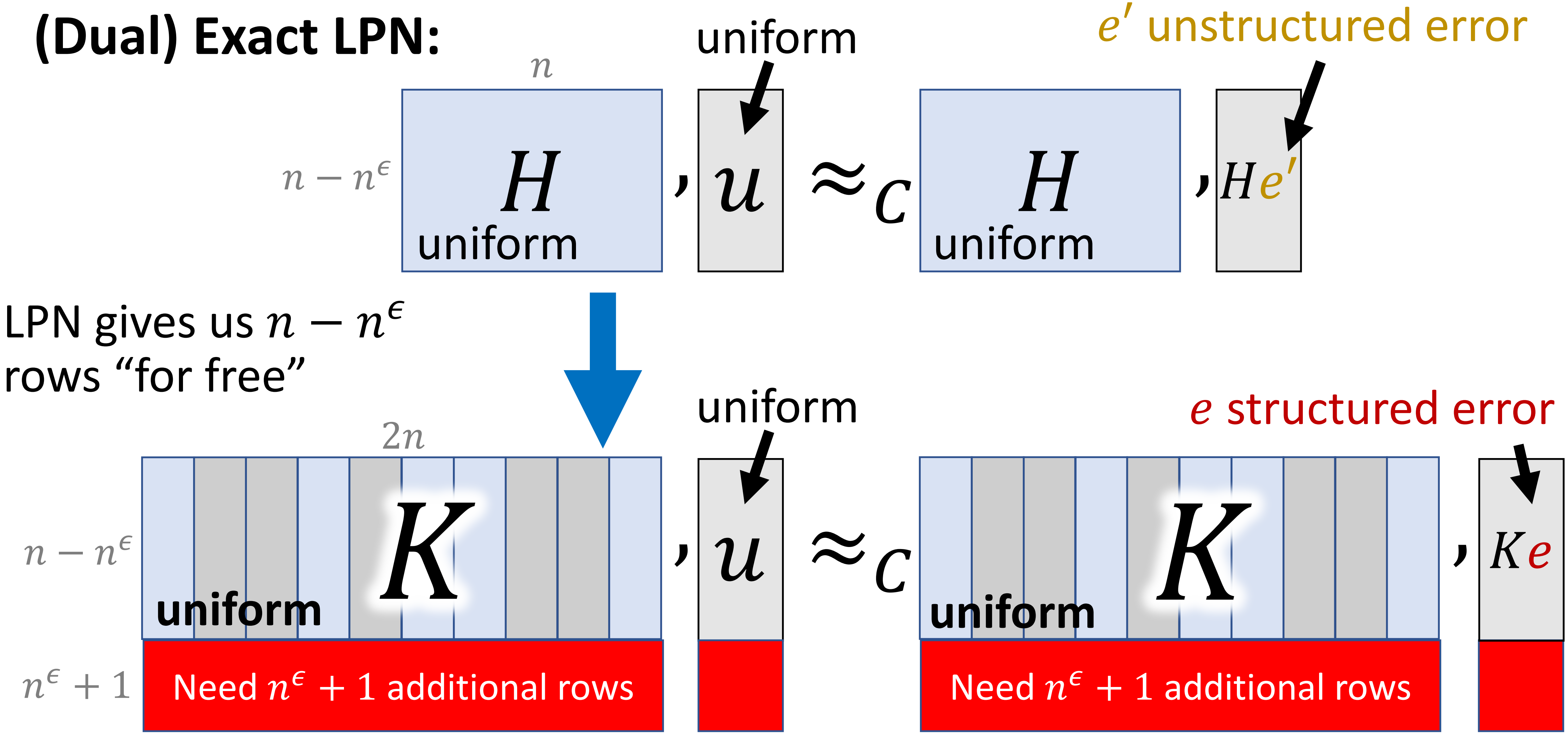
Claim: $\boldsymbol{He'} = \boldsymbol{Ke}$ where $\boldsymbol{e}'$ is unstructured error and $\boldsymbol{e}$ is structured error.

(Dual) Exact LPN:

LPN gives us $n - n^{\epsilon}$ rows "for free"

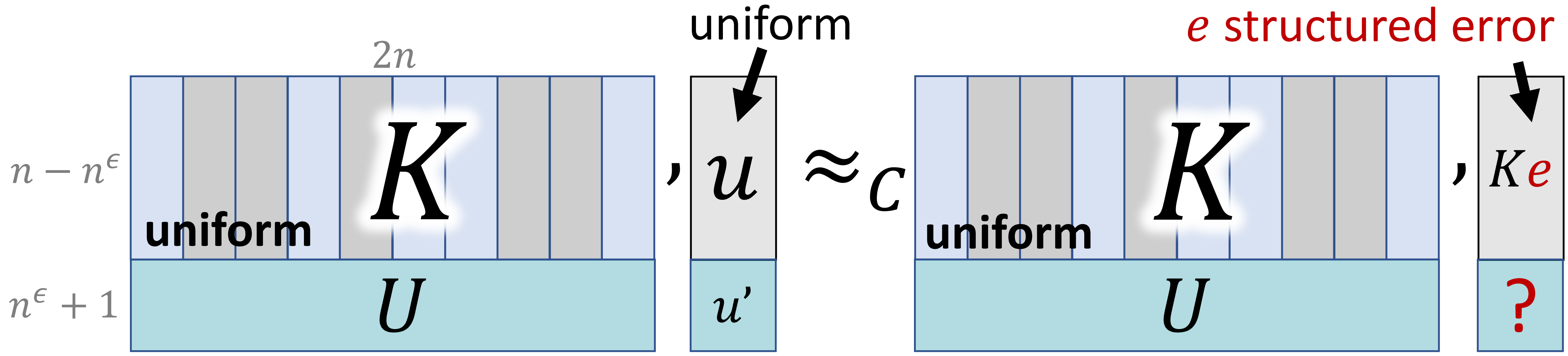


(Dual) Exact LPN:

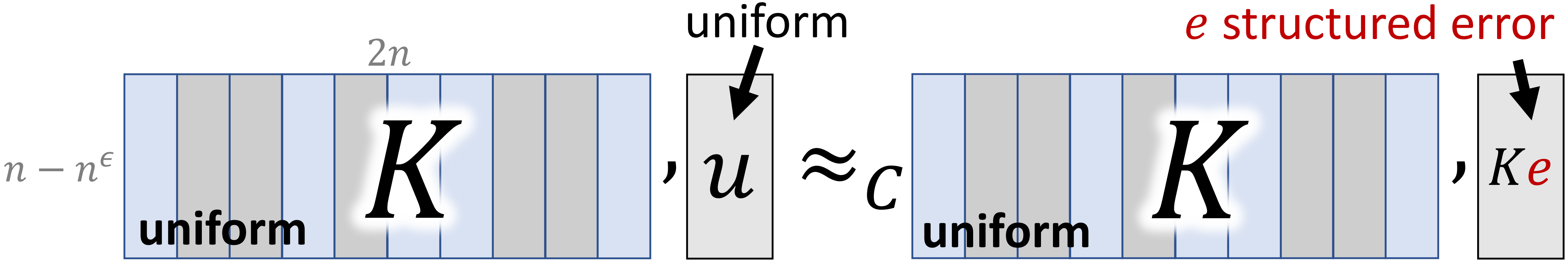


We need $n + 1$ rows for the obfuscation construction.

Issue: If we sample additional rows U uniformly at random, we can't fill in Ue without e.

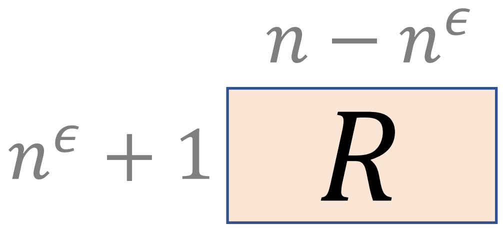


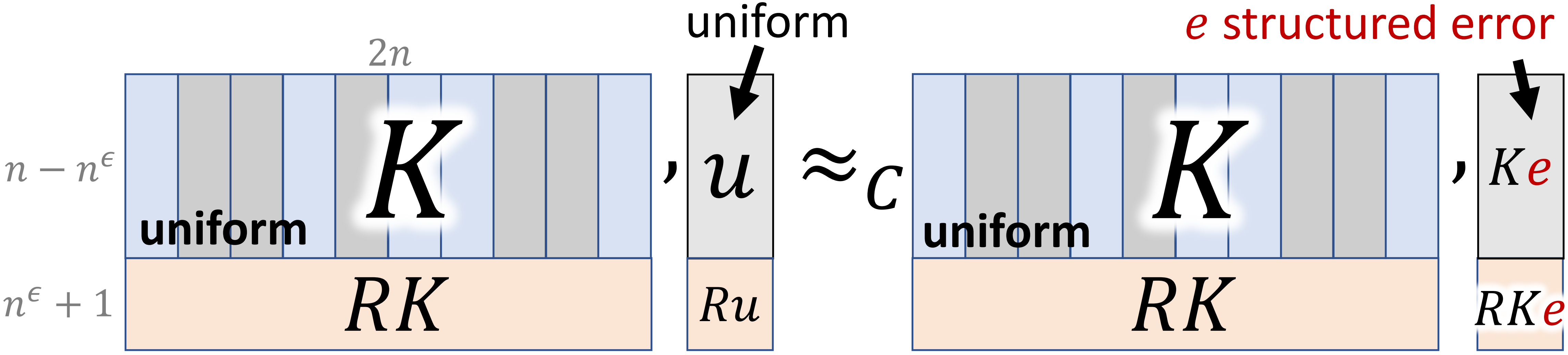
Observation: We **know** $K_i e$ for any row K_i of K.
So we can use random linear combinations of rows of K.



Observation: We **know** $K_i e$ for any row K_i of K.
So we can use random linear combinations of rows of K.

Sample random matrix R:

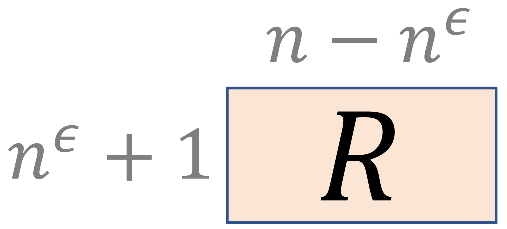


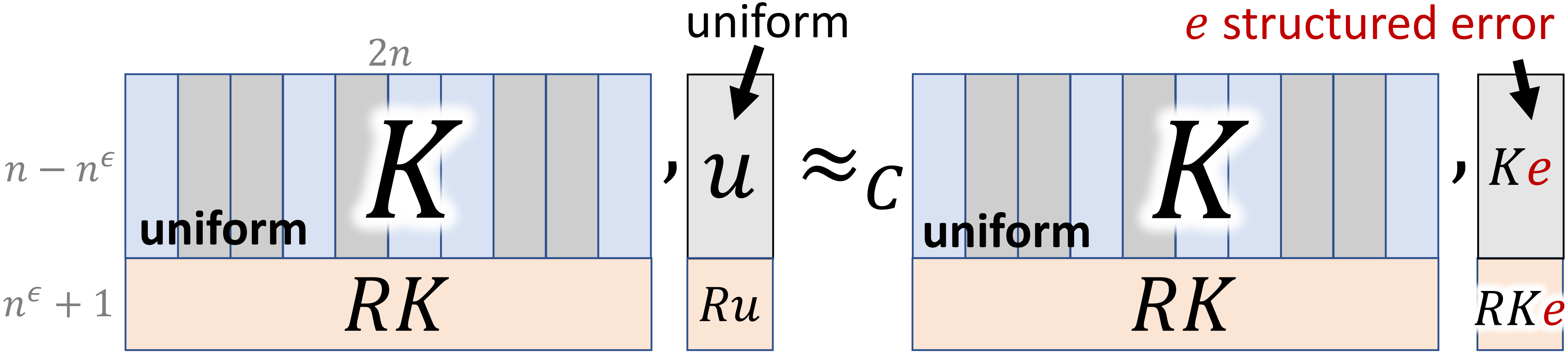


So are we done?

Observation: We **know** $K_i e$ for any row K_i of K.
So we can use random linear combinations of rows of K.

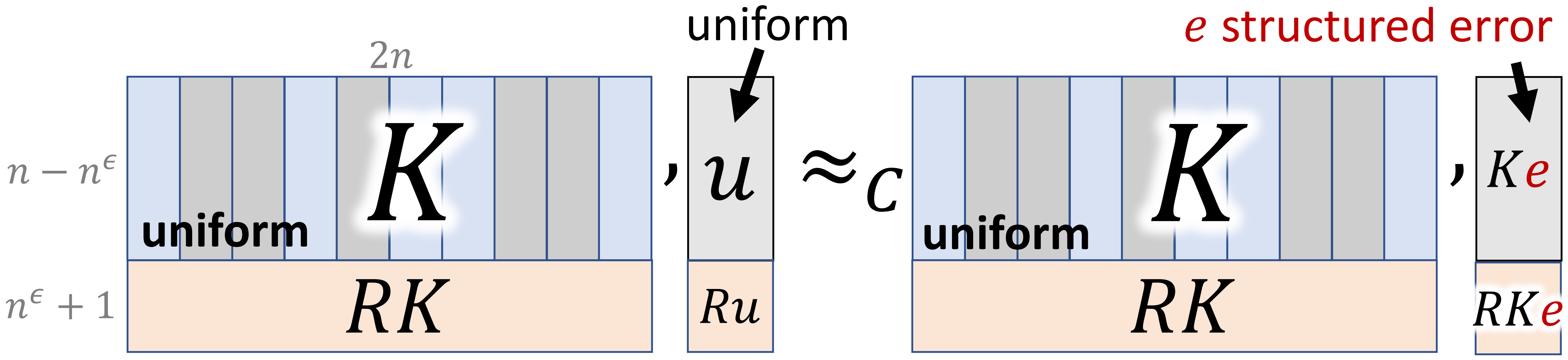
Sample random matrix R:





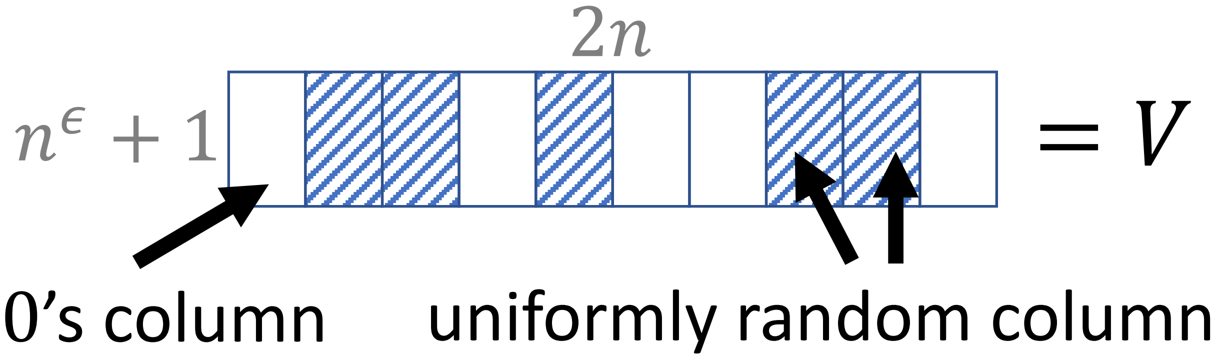
The matrix isn't random!
(rank is at most $n - n^{\epsilon}$)

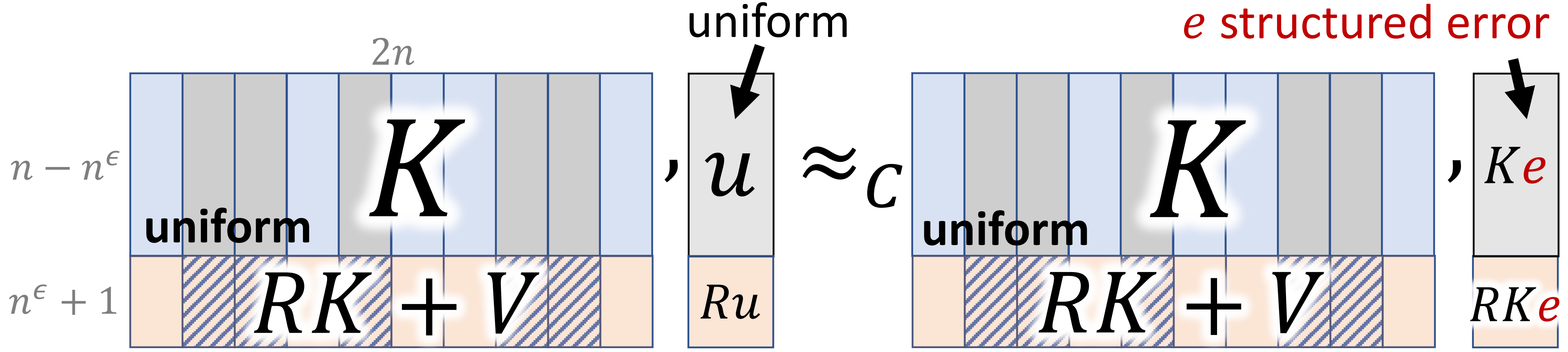
One last idea: we know half the entries of e since we implicitly "inserted" n zeros.



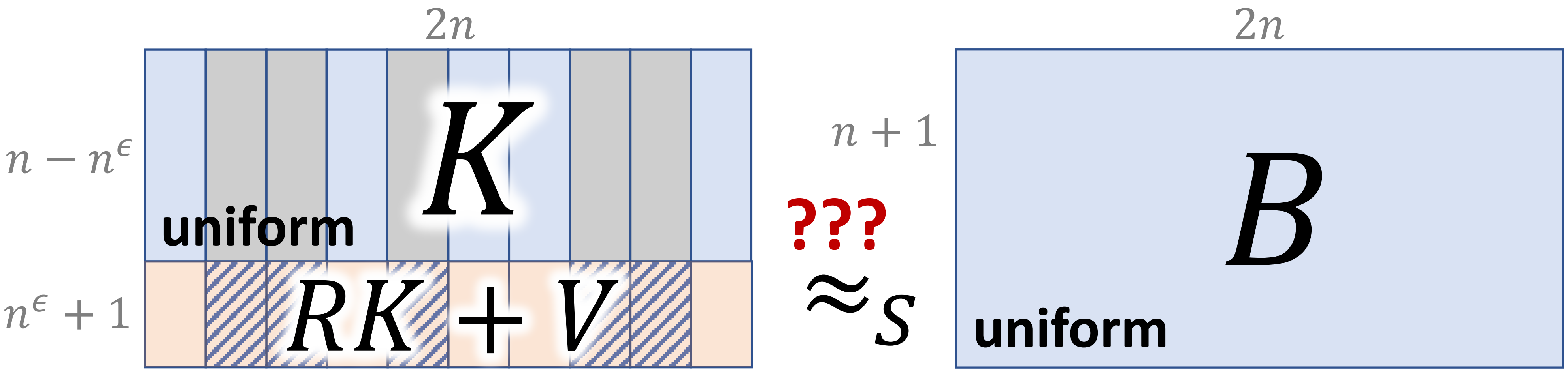
One last idea: we know half the entries of e since we implicitly "inserted" n zeros.

Sample matrix V with n uniformly random non-zero columns coinciding with n known zero entries of e. (i.e. $Ve = 0$)

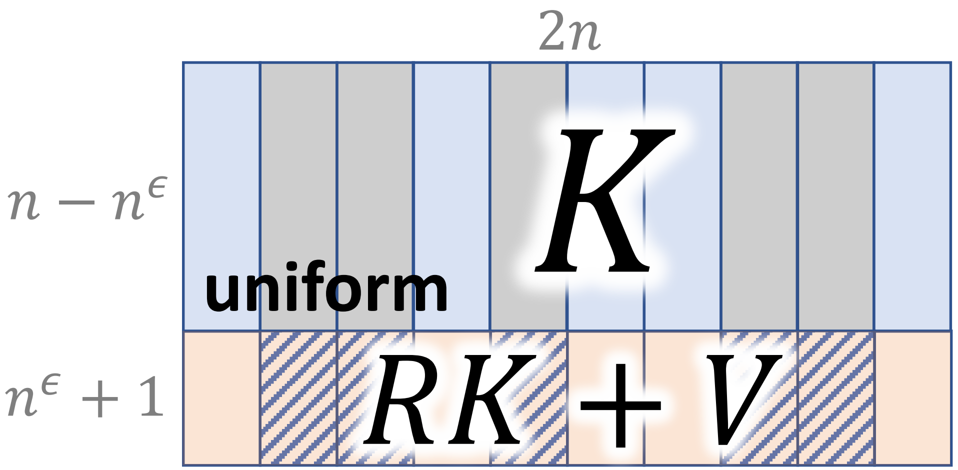




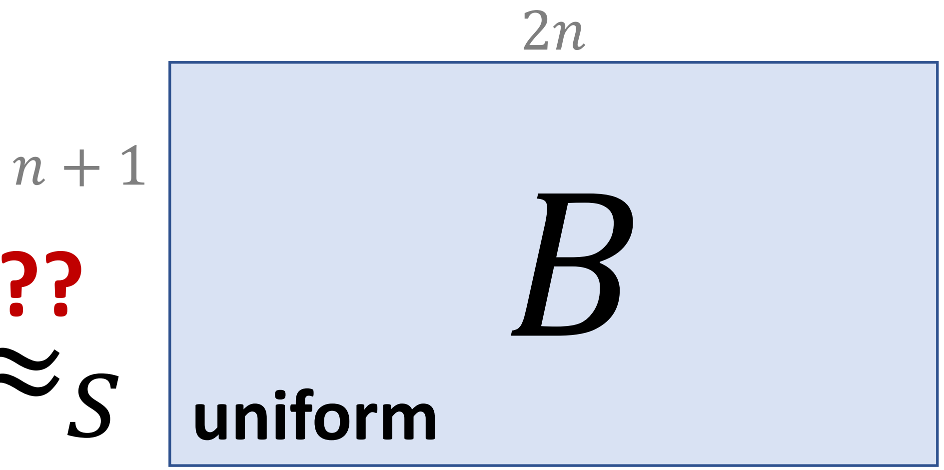
Does $(K, RK + V)$ look uniformly random?

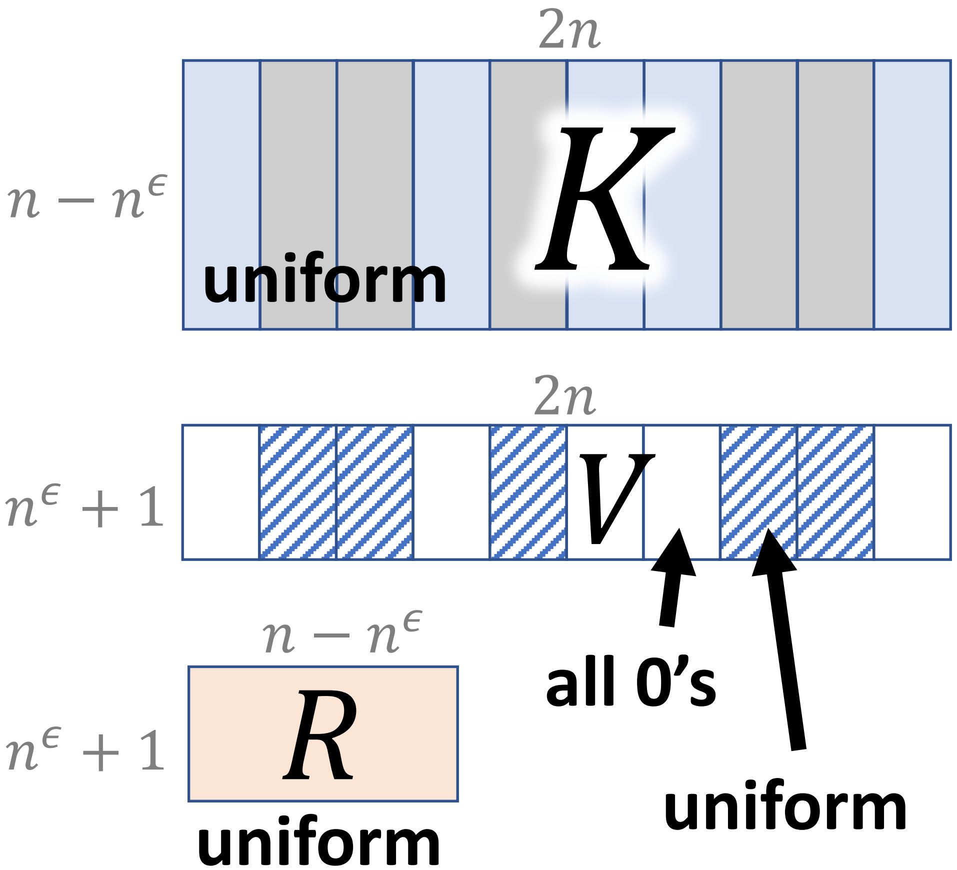


Does $(K, RK + V)$ look uniformly random?



???
$\approx_S$





$\boxed{\text{matrices over } \mathbb{F}_q, \log q = n^{\delta}}$

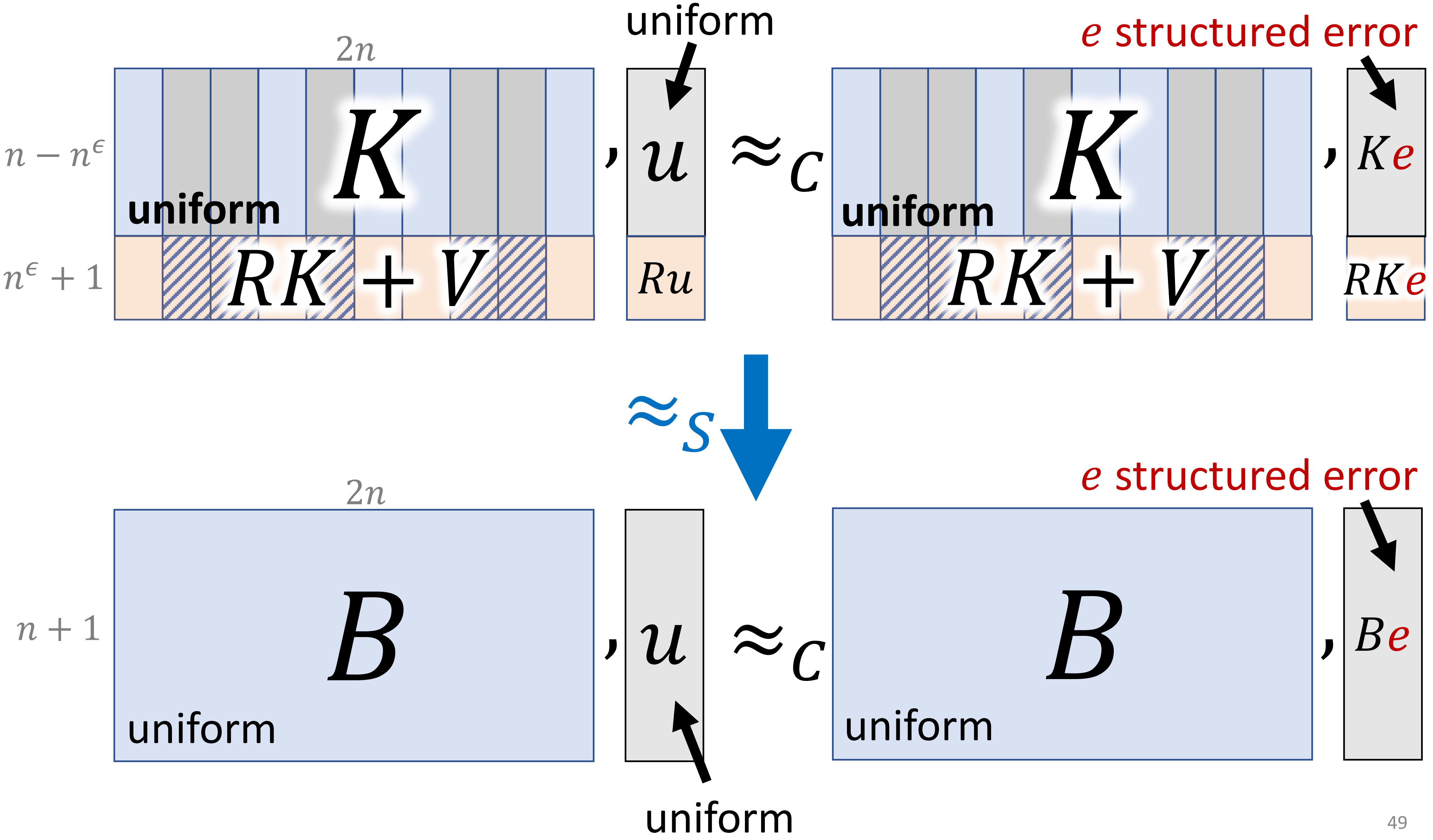
Heuristic Argument: Entropy Counting

$$\boldsymbol{H}_{\infty}(K) + \boldsymbol{H}_{\infty}(R) + \boldsymbol{H}_{\infty}(V)$$

$$= n^{\delta}(\#\text{ entries in } K) + n^{\delta}(\#\text{ entries in } R) + n^{\delta}(\#\text{ nonzero entries in } V) + n$$

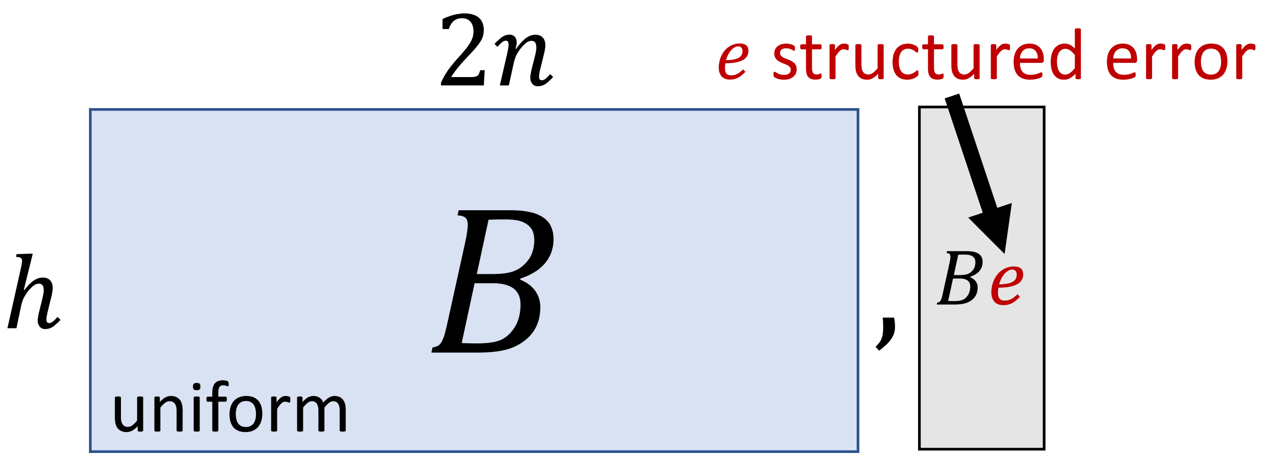
$$= \boldsymbol{H}_{\infty}(B) - n^{2\epsilon+\delta} + n$$

If $2\epsilon + \delta < 1$, LHL yields $(K, RK + V)$ statistically close to uniform.



uniform
e structured error
$2n$
$n - n^\epsilon$
$n^\epsilon + 1$
K
uniform
$RK + V$
$, u$
Ru
$\approx_C$
K
uniform
$RK + V$
$, Ke$
RKe
$\approx_S$
e structured error
$2n$
$n + 1$
B
uniform
$, u$
$\approx_C$
B
uniform
$, Be$
uniform

Another Perspective: Structured Error LPN

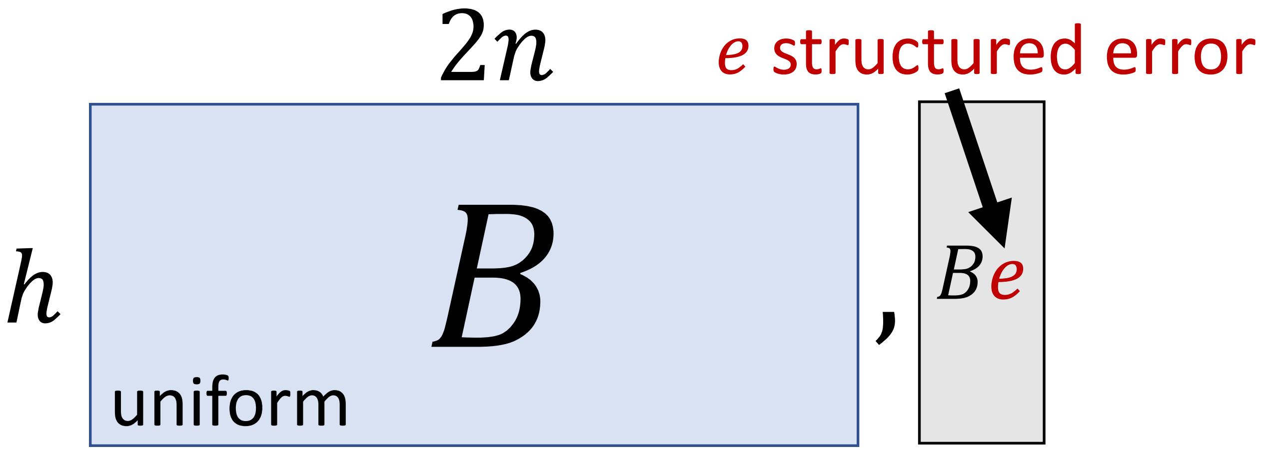


For what h is (B, Be) pseudorandom?

- Pseudorandom if $h = n - n^{\epsilon}, \epsilon < 1$ (perfectly equivalent to Exact LPN)

- [AroraGe12] Can solve for e if $h = 2n - n^{\delta}, \delta < 1/2$

Another Perspective: Structured Error LPN



For what h is (B, Be) pseudorandom?

- Pseudorandom if $h = n - n^{\epsilon}, \epsilon < 1$ (perfectly equivalent to Exact LPN)
- **[This work]** Pseudorandom if $h = n + n^{\gamma}, \gamma < 1/2$ (statistically equivalent to Exact LPN)
- [AroraGe12] Can solve for e if $h = 2n - n^{\delta}, \delta < 1/2$

Conclusion

- In the GGM: obfuscate conjunctions by encoding in a vector and multiplying by a structured matrix.
- If we multiply by a random matrix, we can avoid groups and rely on LPN.

In the paper:

- An **information theoretic** conjunction obfuscator consisting of a sequence of matrices; evaluation is done by taking a subset-sum of matrices and computing the **determinant**.

Thank You!

ePrint: ia.cr/2018/936
slides: cs.princeton.edu/~fermim/talks/crypto-day.pdf