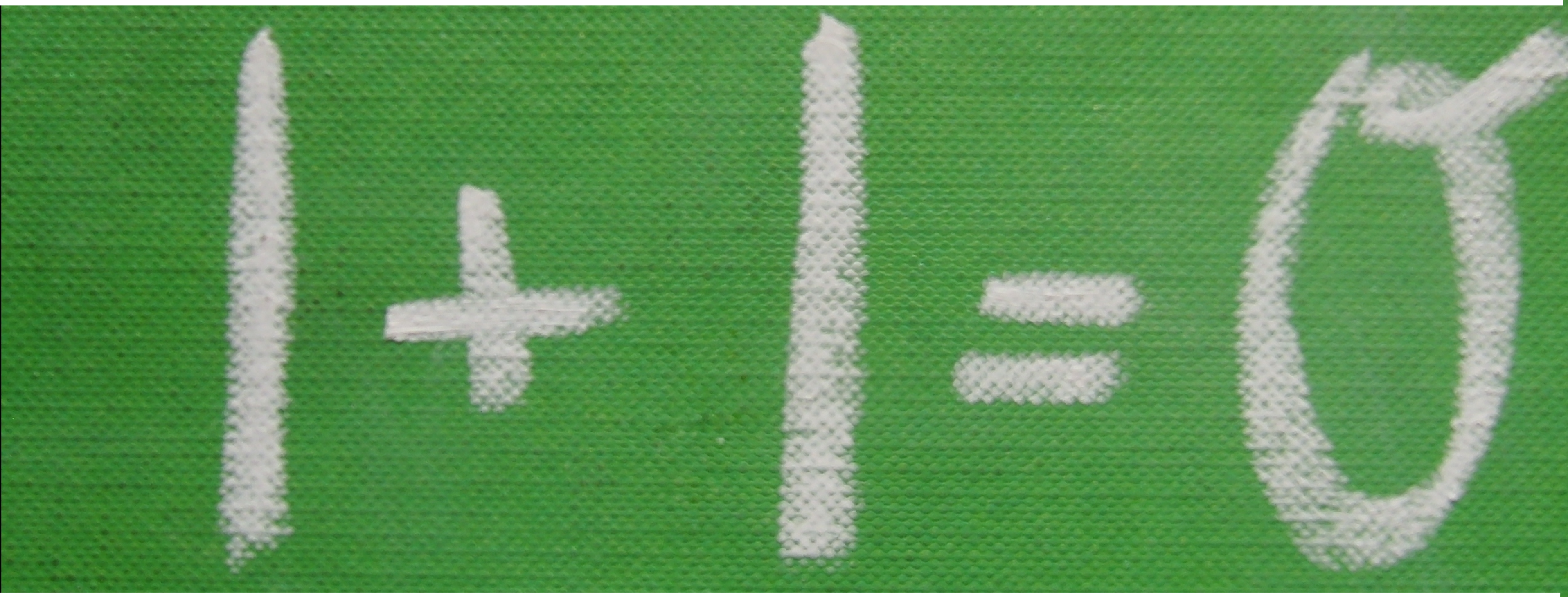
Improved Information Set Decoding



Alexander Meurer, Ruhr-Universität Bochum

CBC Workshop 2012, Lyngby

The Asymptotic Playground

- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices

The Asymptotic Playground

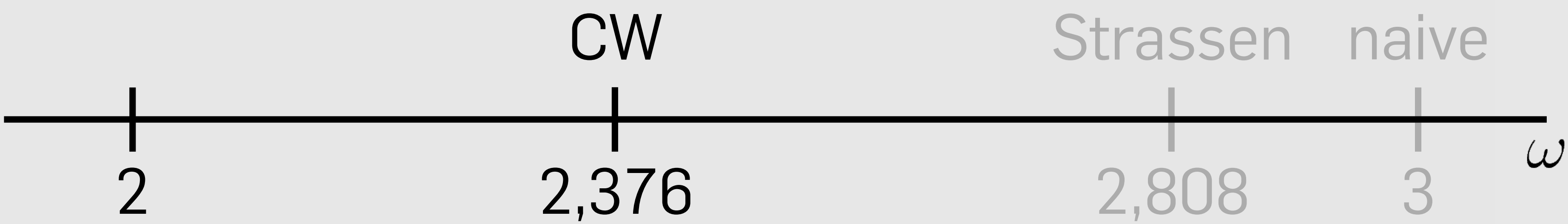
- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices

The Asymptotic Playground

- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices

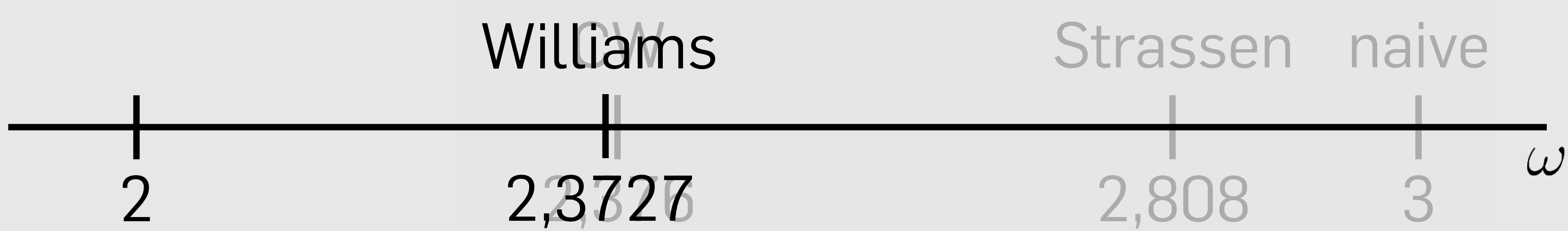
The Asymptotic Playground

- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices



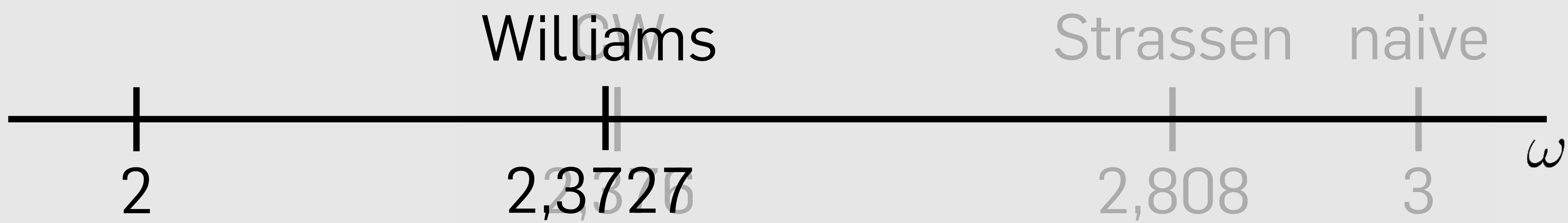
The Asymptotic Playground

- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices



The Asymptotic Playground

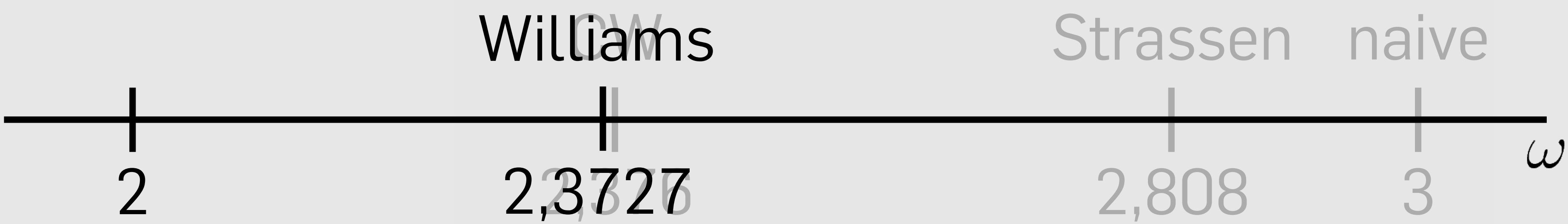
- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices



- Strassen still performs best in practice (for reasonable n)

The Asymptotic Playground

- We are interested in asymptotically fastest algorithms
- Prominent example: Matrix multiplication
- Measure runtime as $\mathcal{O}\left(n^{\omega}\right)$ for n x n - matrices



This talk: **recent (asymptotic) progress in ISD**.

Recap Binary Linear Codes

RUB

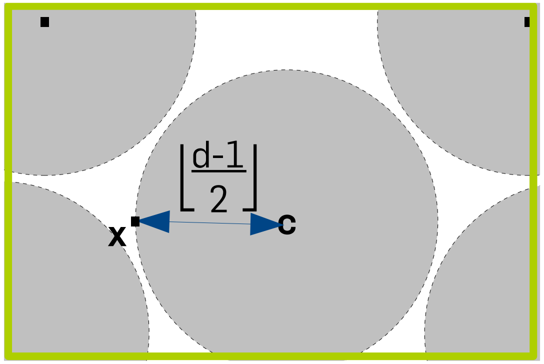
- C = random binary [n,k,d] code
- n = length / k = dimension / d = minimum distance

Bounded Distance Decoding (BDD)

- Given $\mathbf{x} = \mathbf{c}+\mathbf{e}$ with $\mathbf{c} \in C$

  and $w := wt(\mathbf{e}) = \left\lfloor \frac{d-1}{2} \right\rfloor$

- Find $\mathbf{e}$ and thus $\mathbf{c} = \mathbf{x}+\mathbf{e}$



Comparing Running Times

How to compare performance of decoding algorithms

- Running time $T(n,k,d)$
- Fixed code rate $R = k/n$
- For $n \to \infty$, k and d are related via Gilbert-Varshamov bound, thus

$$T(n,k,d) = T(n,k)$$

- Compare algorithms by complexity coefficient $F(k)$, i.e.

$$T(n,k) = 2^{F(k) \cdot n + o(n)}$$

Comparing Running Times

How to compare performanc … s

Minimize F(k)!

- Running time T(n,k,d)
- Fixed code rate R = k/n
- For $n \to \infty$, k and d are related vi … ert-Varshamov bound, thus

$$T(n,k,d) = T \ldots)$$

- Compare algorithms by compl … xity coefficient F(k), i.e.

$$T(n,k) = 2^{F(k) \cdot n + o(n)}$$

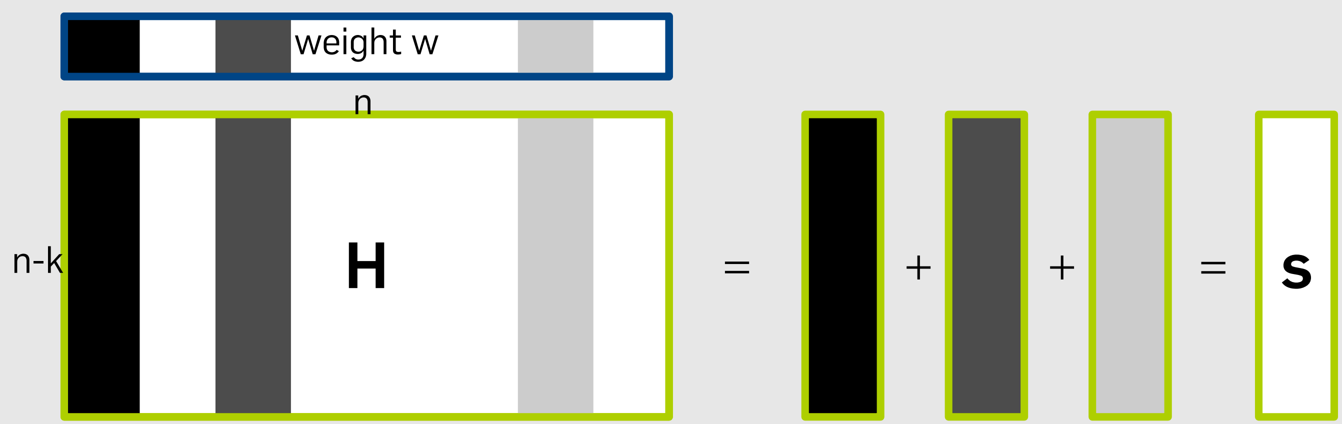
Syndrome Decoding

(BDD) Given $\mathbf{x} = \mathbf{c}+\mathbf{e}$ with $\mathbf{c} \in C$ and $\mathrm{wt}(\mathbf{e})=w$, find $\mathbf{e}$!

- $\mathbf{H}$ = parity check matrix

- Consider syndrome $\mathbf{s} := s(\mathbf{x}) = \mathbf{H}\cdot\mathbf{x} = \mathbf{H}\cdot(\mathbf{c}+\mathbf{e}) = \mathbf{H}\cdot\mathbf{e}$

→ Find linear combination of w columns of **H** matching **s**



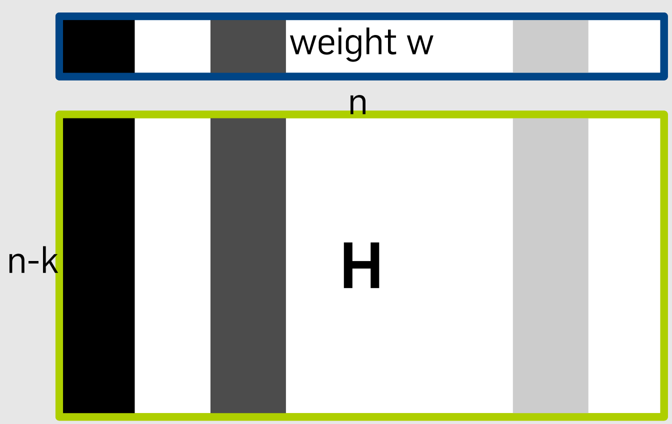
Syndrome Decoding

(BDD) Given $\mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in C$ and $\mathrm{wt}(\mathbf{e}) = w$, find $\mathbf{e}$!

- $\mathbf{H}$ = parity check matrix

- Consider syndrome $\mathbf{s} := s(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x} = \mathbf{H} \cdot (\mathbf{c} + \mathbf{e}) = \mathbf{H} \cdot \mathbf{e}$

→ Find linear combination of w columns of **H** matching **s**

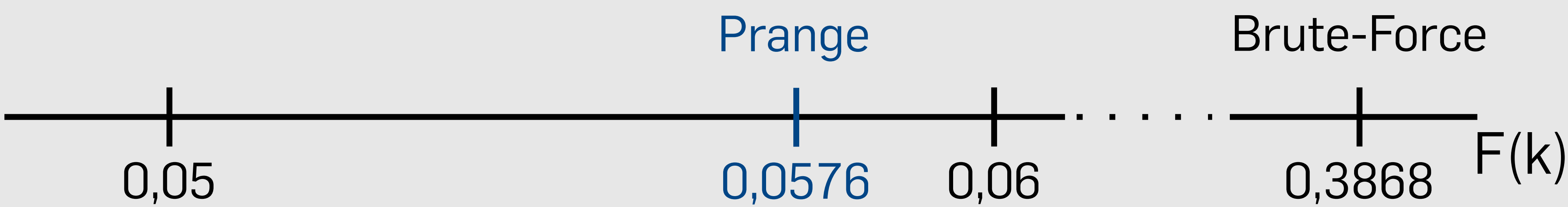


Brute-Force complexity

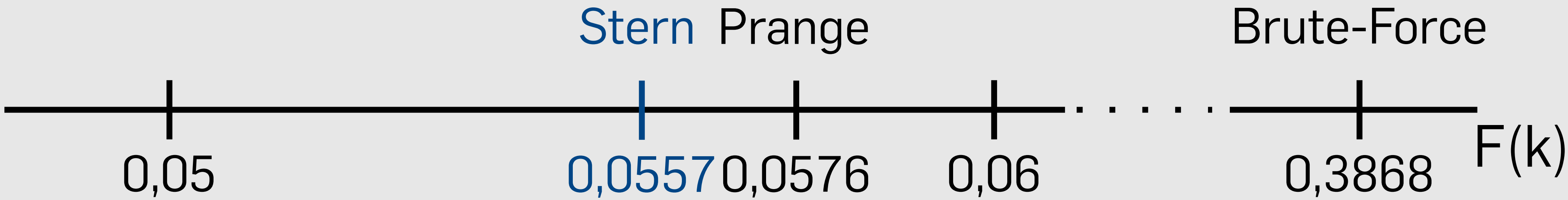
$$T(n,k,d) = \binom{n}{w}$$

Complexity Coefficients (BDD)

Complexity Coefficients (BDD)



Complexity Coefficients (BDD)



Some Basic Observations for BDD

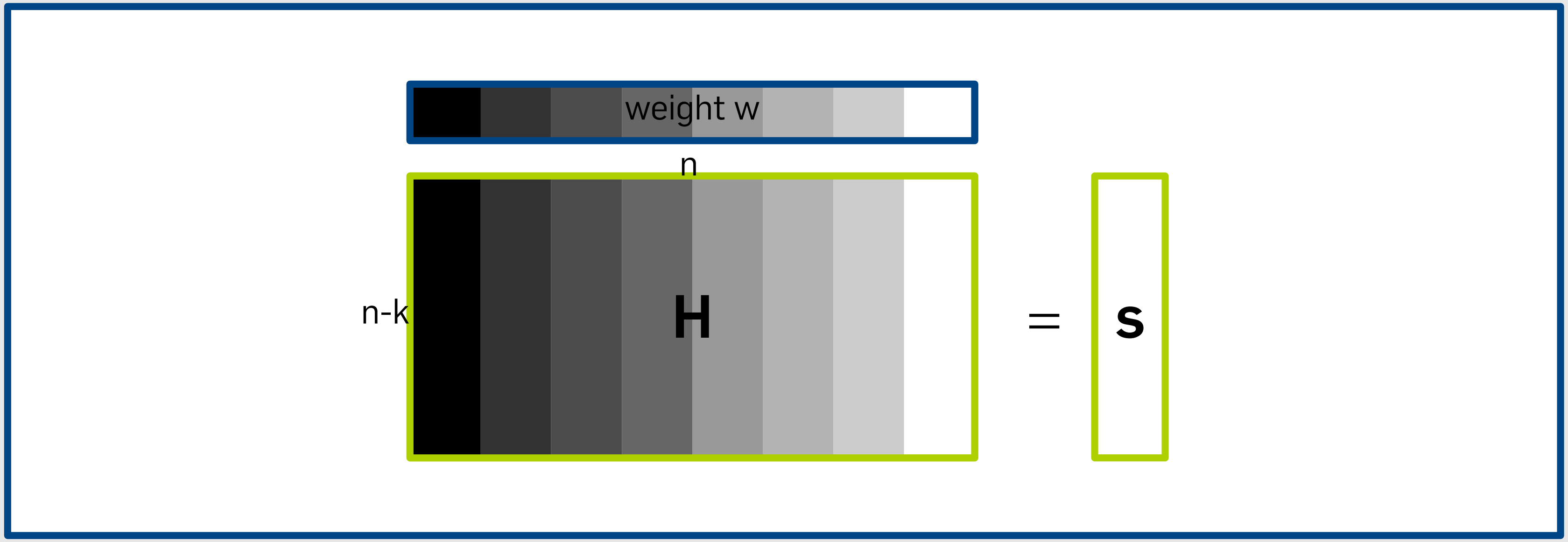
Allowed (linear algebra) transformations

- Permuting the columns of **H** does not change the problem

Some Basic Observations for BDD

Allowed (linear algebra) transformations

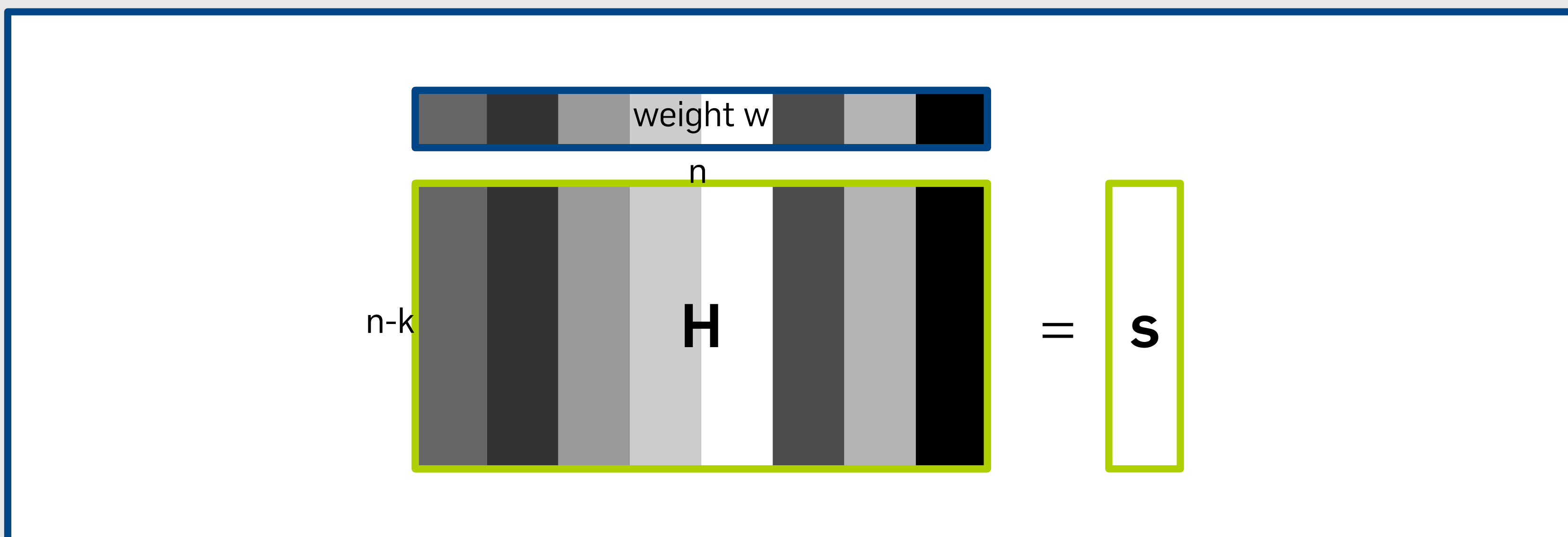
- Permuting the columns of **H** does not change the problem



Some Basic Observations for BDD

Allowed (linear algebra) transformations

- Permuting the columns of **H** does not change the problem



Some Basic Observations for BDD

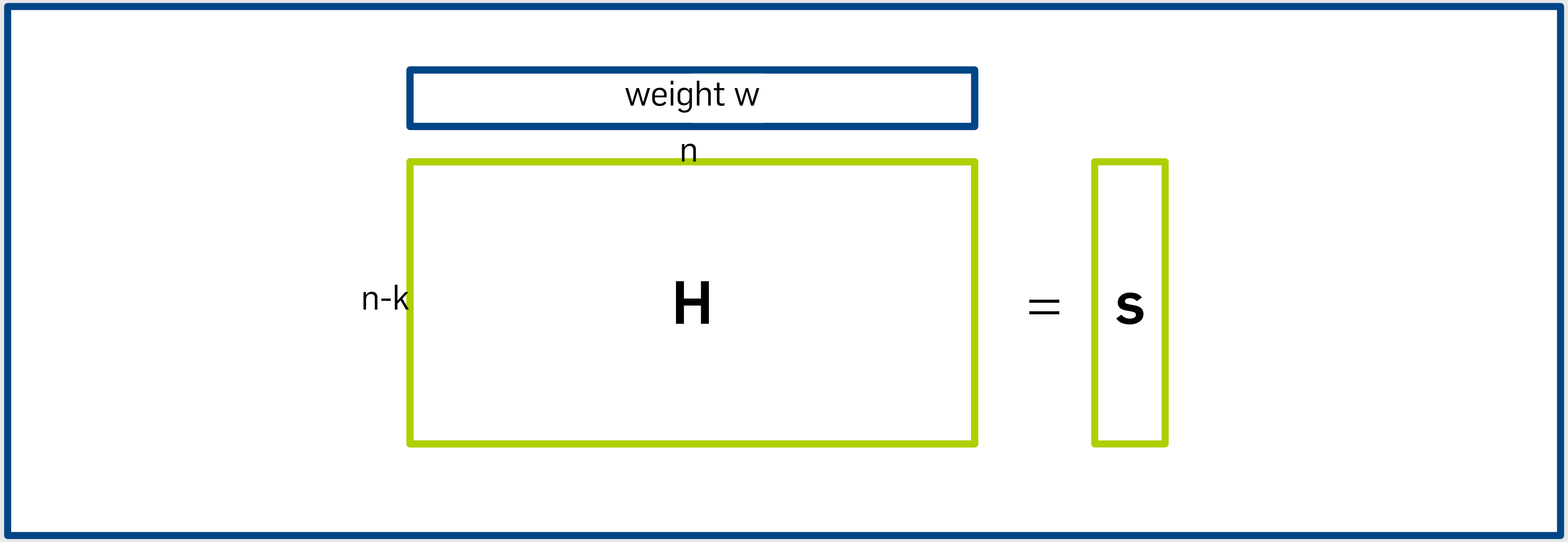
Allowed (linear algebra) transformations

- Permuting the columns of **H** does not change the problem
- Elementary row operations on **H** do not change the problem

Some Basic Observations for BDD

Allowed (linear algebra) transformations

- Permuting the columns of **H** does not change the problem
- Elementary row operations on **H** do not change the problem

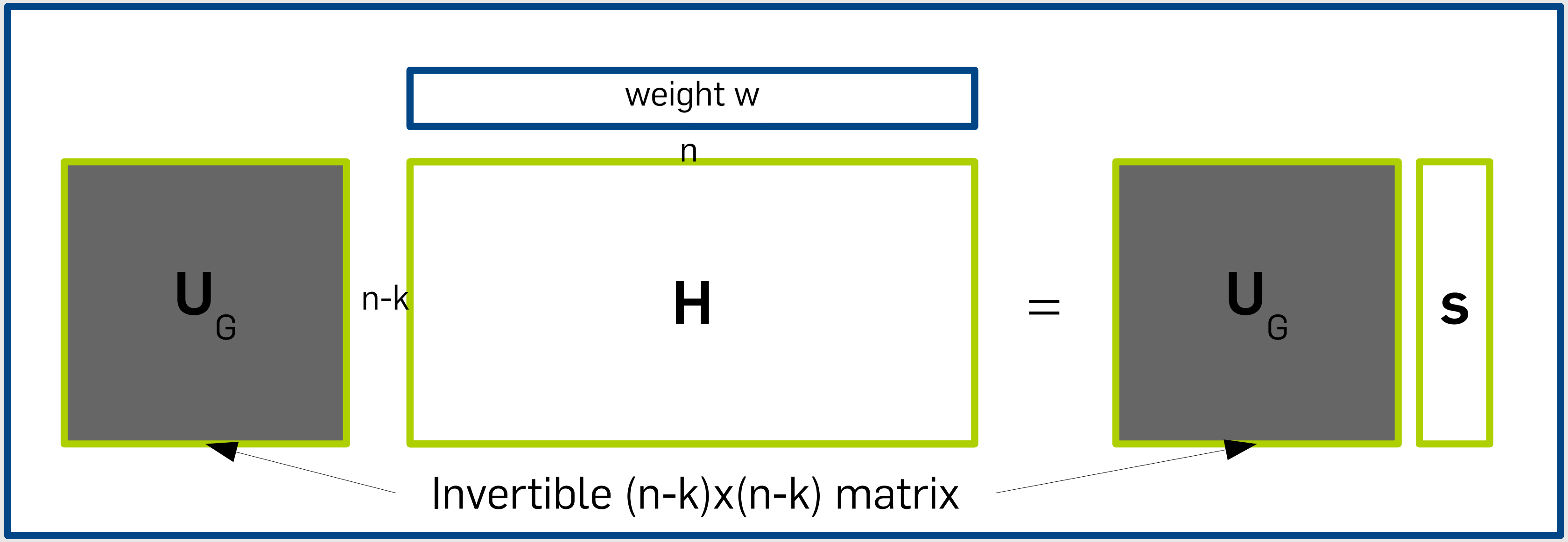


Some Basic Observations for BDD

RUB

Allowed (linear algebra) transformations

- Permuting the columns of **H** does not change the problem
- Elementary row operations on **H** do not change the problem



Randomized quasi-systematic form

RUB

- Work on randomly column-permuted version of **H**
- Transform **H** into quasi-systematic form

$$\mathbf{H} = \begin{array}{cc} k+l & n-k-l \\ \left[\begin{array}{c|c} \mathbf{q}_1, \ldots, \mathbf{q}_{k+l} & \mathbf{0} \\ \hline \mathbf{Q}' & \mathbf{I}_{n-k-l} \end{array}\right] \end{array} \begin{array}{l} \left.\vphantom{\mathbf{q}_1}\right\} l \text{ rows} \\ \vphantom{\mathbf{Q}'} \end{array}$$

First used in generalized ISD framework of [FS09]

Information Set Decoding

''Reducing the brute-force search space by linear algebra.''

The ISD Principle

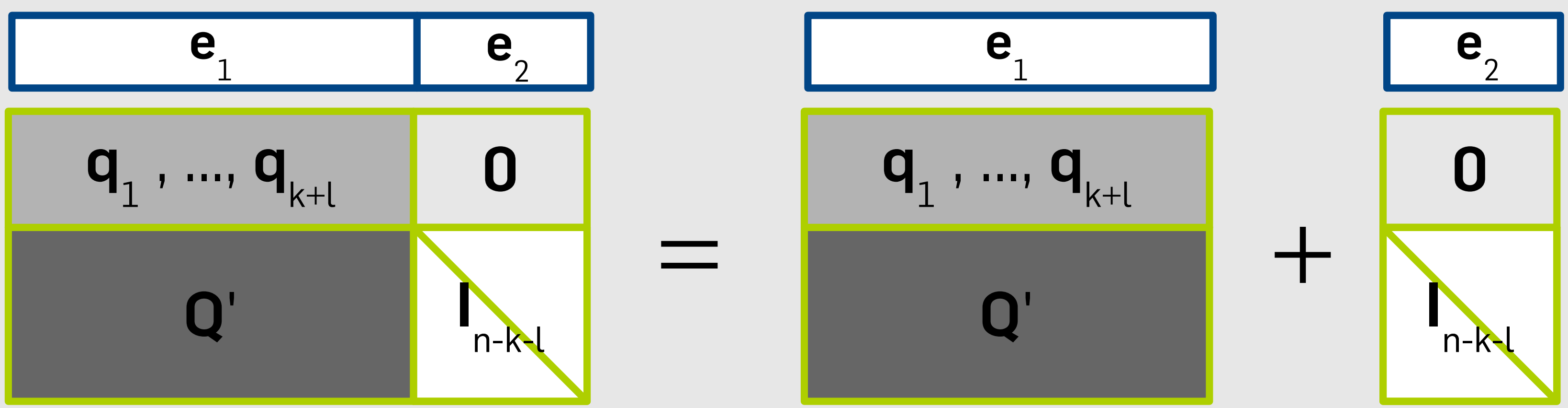
RUB

- Structure of **H** allows to divide $\mathbf{e} = \begin{array}{cc} \scriptstyle k+l & \scriptstyle n-k-l \\ \boxed{\quad \mathbf{e}_1 \quad} & \boxed{\mathbf{e}_2} \end{array}$

$$
\begin{array}{cc}
\boxed{\quad \mathbf{e}_1 \quad} & \boxed{\mathbf{e}_2} \\
\mathbf{q}_1, \ldots, \mathbf{q}_{k+l} & \mathbf{0} \\
\mathbf{Q}' & \mathbf{I}_{n-k-l}
\end{array}
$$

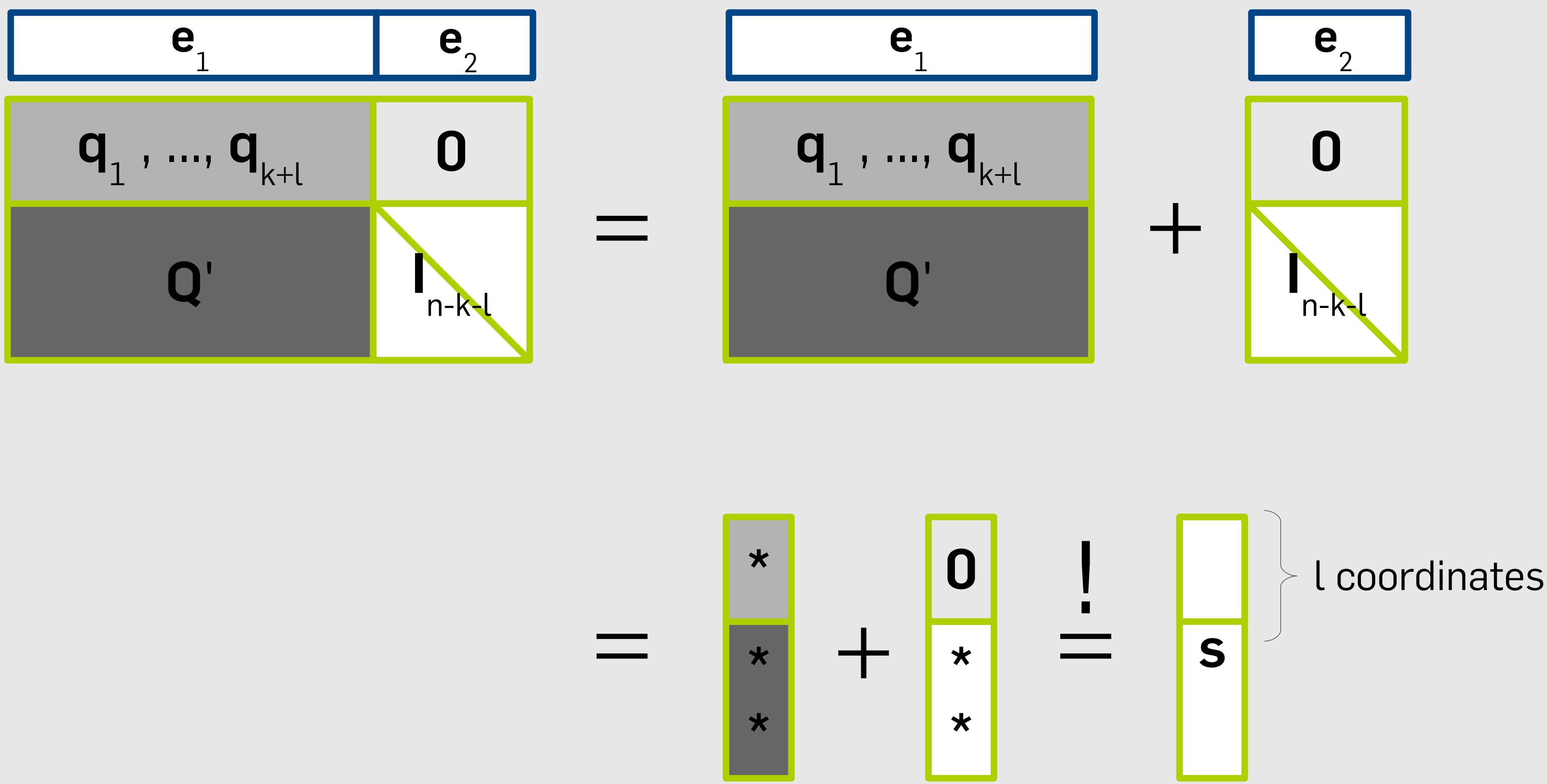
The ISD Principle

- Structure of **H** allows to divide $\mathbf{e} = \boxed{\mathbf{e}_1 \mid \mathbf{e}_2}$



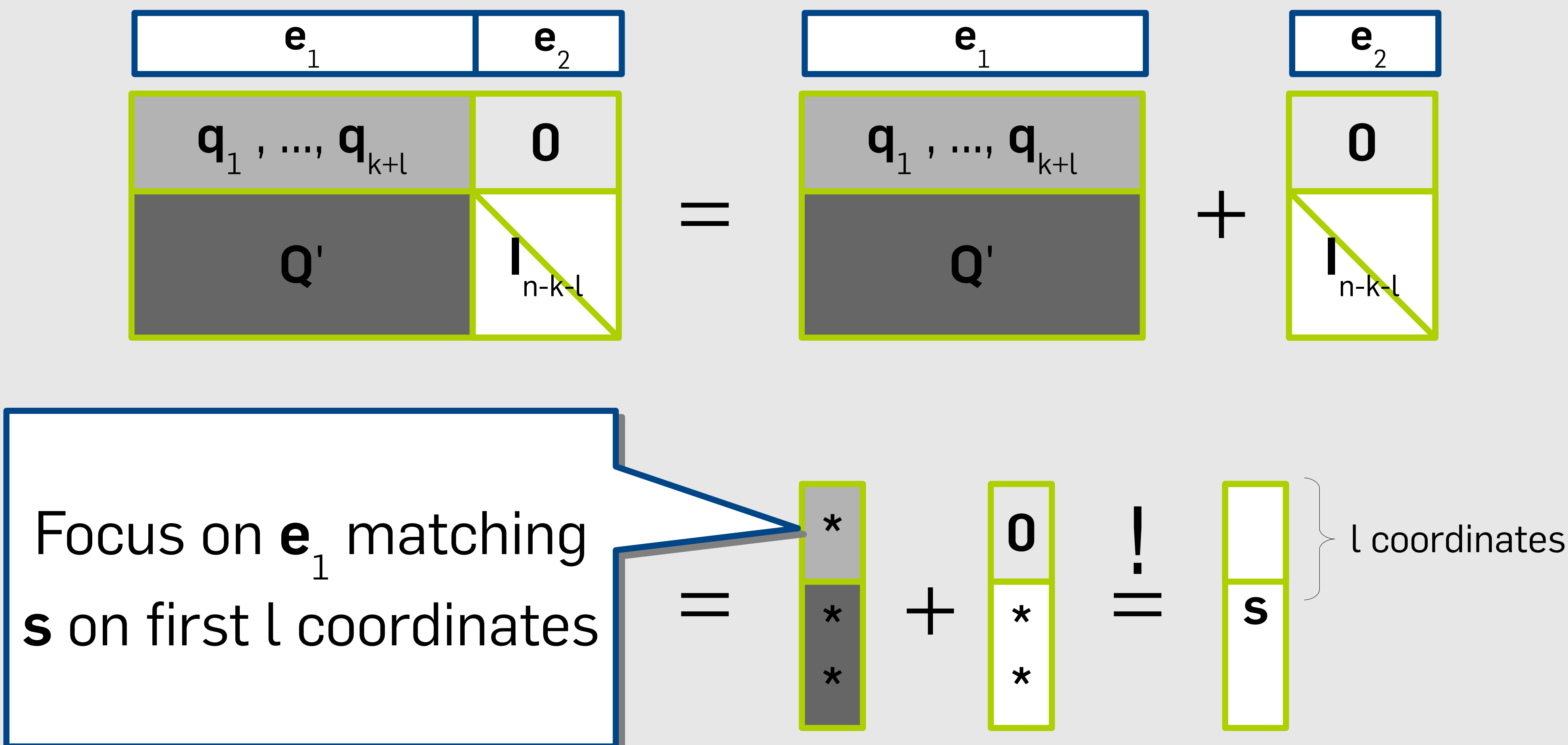
The ISD Principle

- Structure of **H** allows to divide $\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix}$



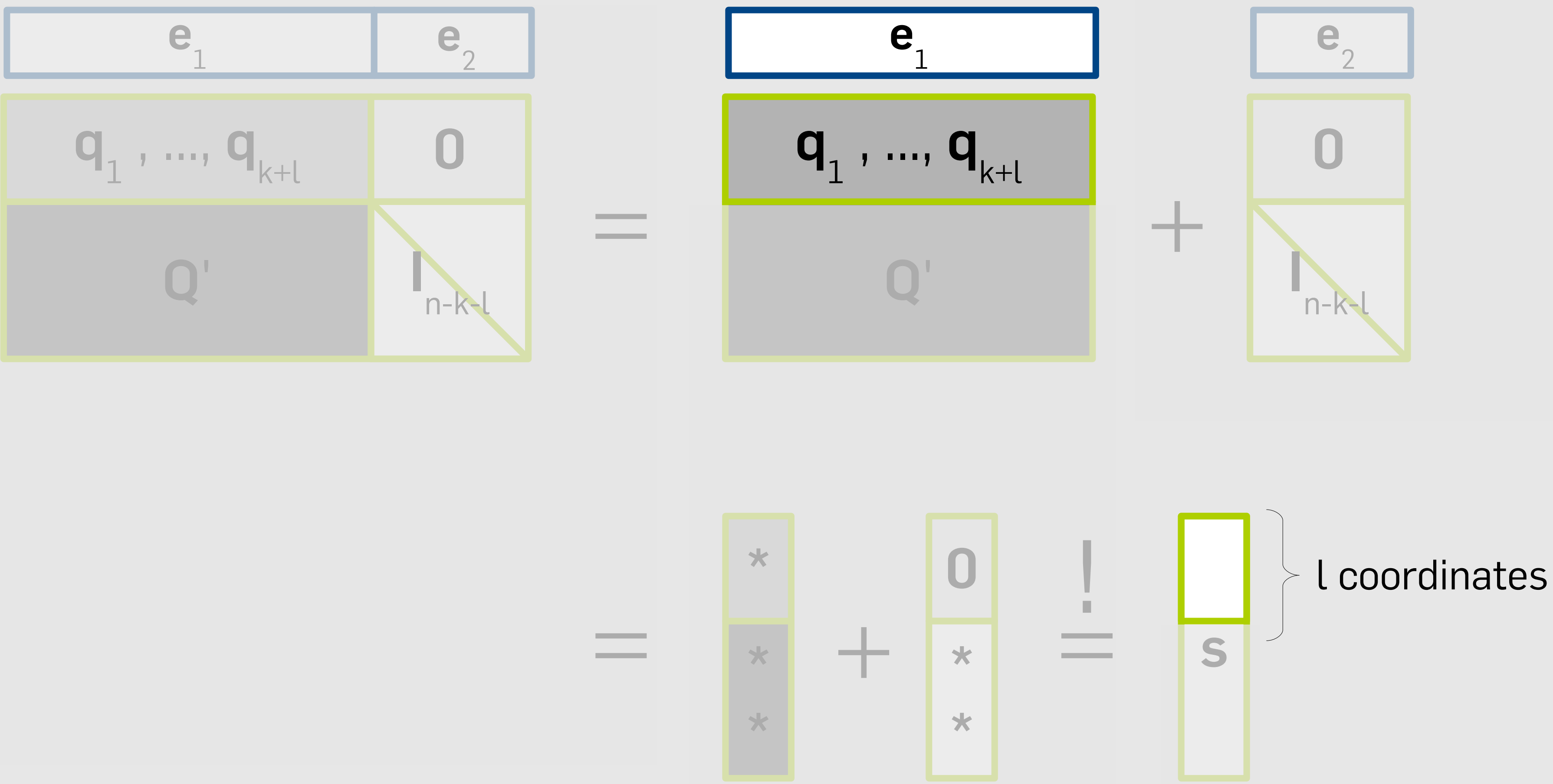
The ISD Principle

- Structure of **H** allows to divide $\mathbf{e} = [\,\mathbf{e}_1 \mid \mathbf{e}_2\,]$



The ISD Principle

Find all $\mathbf{e}_1$ of weight p matching $\mathbf{s}$ on first l coordinates



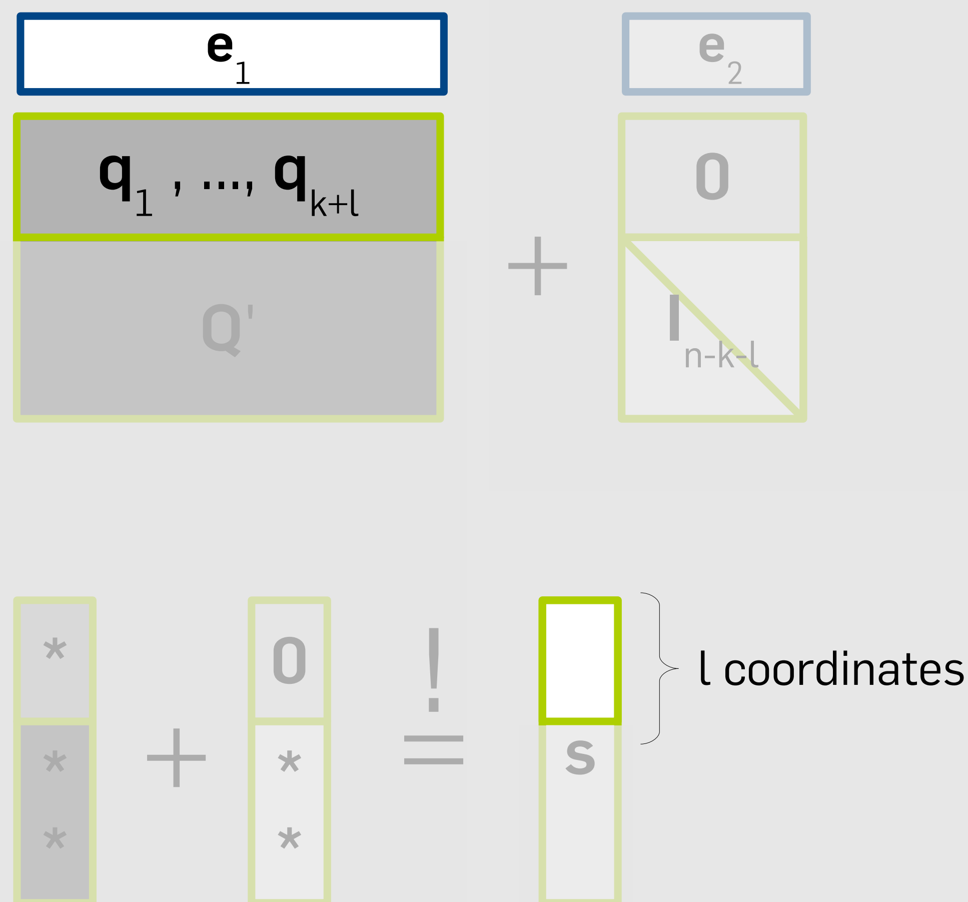
The ISD Principle

Find all $\mathbf{e}_1$ of weight p matching **s** on first l coordinates

- Method only recovers particular error patterns

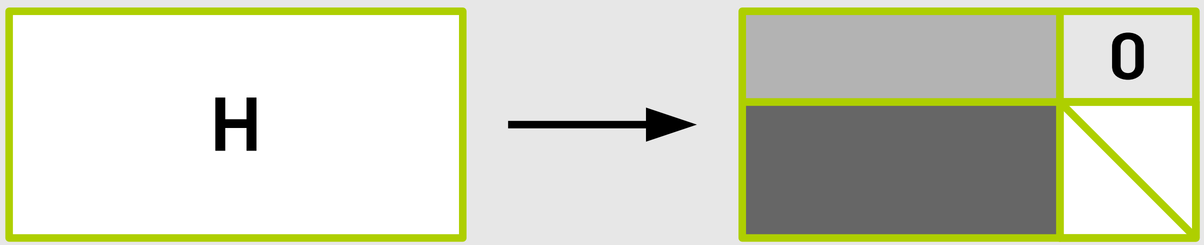
k+l	n-k-l
$\mathbf{e}_1$	$\mathbf{e}_2$
p	w-p

- If no solution found:
  → Rerandomize **H**



The ISD Principle

- 1st step (randomization): Compute „fresh“ random quasi-systematic form of **H**



- 2nd step (search): Try to find a solution **e** amongst all

The ISD Principle

- 1st step (randomization): Compute „fresh“ random quasi-systematic form of **H**

0

$$T = \Pr[\text{„good rand.“}]^{-1} * T[\text{search}]$$

k+l	n-k-l
$\mathbf{e}_1$	$\mathbf{e}_2$
p	w-p

with

$\mathbf{e}_1$

$\mathbf{q}_1, \ldots, \mathbf{q}_{k+l}$ $=$ $\mathbf{s}$

The ISD Search Step (Notation)

- Find vector $\boxed{\mathbf{e}_1}$ of weight p with

$$\boxed{\mathbf{e}_1} \; \boxed{\mathbf{q}_1, \ldots, \mathbf{q}_{k+l}} = \boxed{\mathbf{s}}$$

The ISD Search Step (Notation)

- Find vector $\boxed{\mathbf{e}_1}$ of weight p with

$$\boxed{\mathbf{e}_1}$$
$$\boxed{\mathbf{q}_1, \ldots, \mathbf{q}_{k+l}} = \boxed{\mathbf{s}}$$

- Find selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

The ISD Search Step (Notation)

- Find vector $\boxed{\mathbf{e}_1}$ of weight p with

$$\boxed{\mathbf{e}_1}\;\boxed{\mathbf{q}_1, \ldots, \mathbf{q}_{k+l}} = \boxed{\mathbf{s}}$$

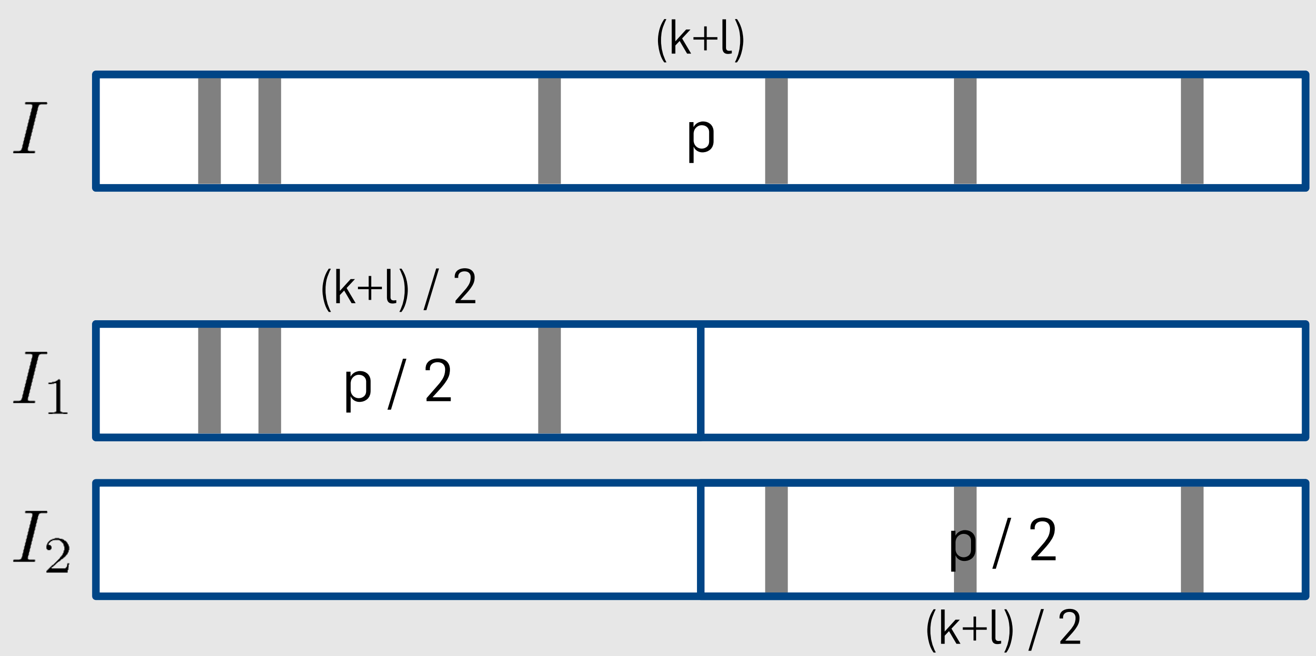
- Find selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

We exploit 1+1=0 to find e_1 more efficiently!

A Meet-in-the-Middle Approach

Find a selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

- Disjoint partition $I = I_1 \dot\cup I_2$ into left and right half

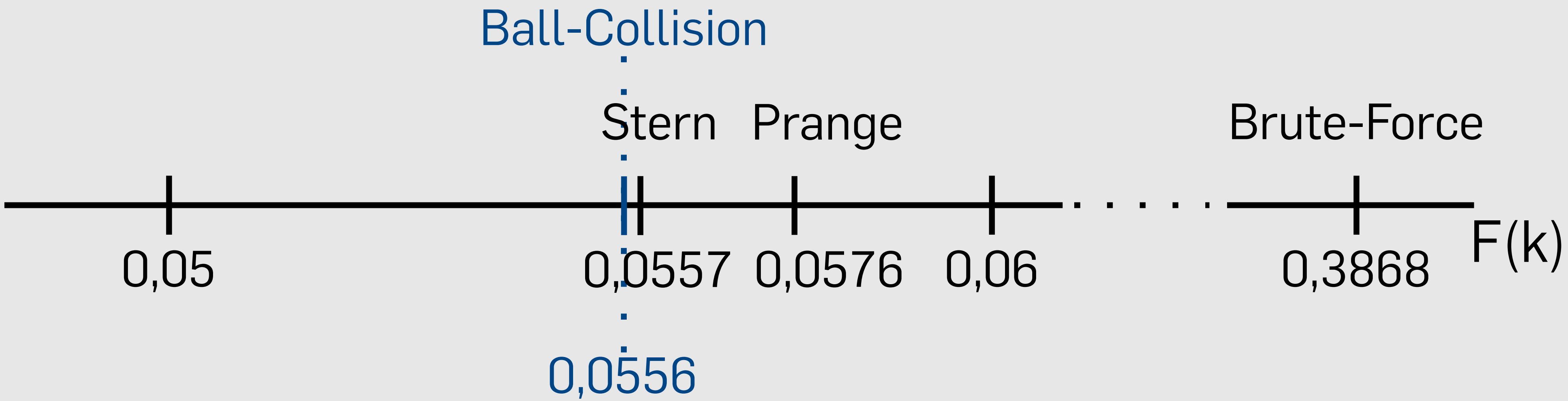


A Meet-in-the-Middle Approach

Find a selection $I \subset [1,\ldots,k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

- To find $I = I_1 \dot{\cup} I_2$ run a Meet-in-the-Middle algorithm based on $\sum_{i \in I_1} q_i = \sum_{j \in I_2} q_j + s$
- Same F(k) as recent Ball-Collision decoding [BLP11] as shown in [MMT11]

Complexity Coefficients (BDD)



The Representation Technique [HGJ10]

How to find a needle N in a haystack H...

- Expand H into larger stack H'
- Expanding H' introduces r many representations $N_1, \ldots, N_r$
- Examine a 1/r – fraction of H' to find one N_i

The Representation Technique [HGJ10]

How to find a needle N in a haystack H...

- Expand H into larger stack H'
- Expanding H' introduces r many representations $N_1, \ldots, N_r$
- Examine a 1/r – fraction of H' to find one N_i

Technicality: Find a way to examine a 1/r – fraction of H' without completely constructing it beforehand

Back to the MitM Approach

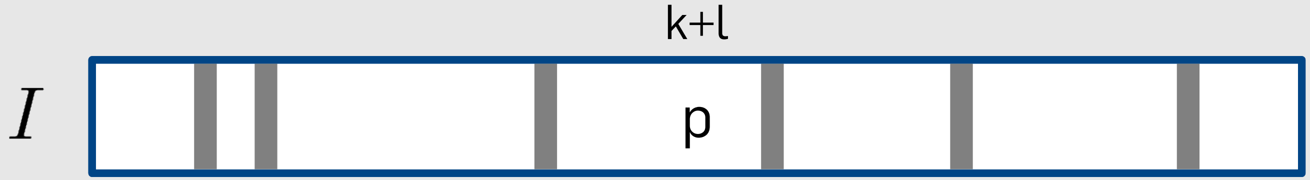
- The disjoint partition forces a unique solution
- Needle = unique 

- Haystack = all vectors 

Using Representations [MMT11]

Find a selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

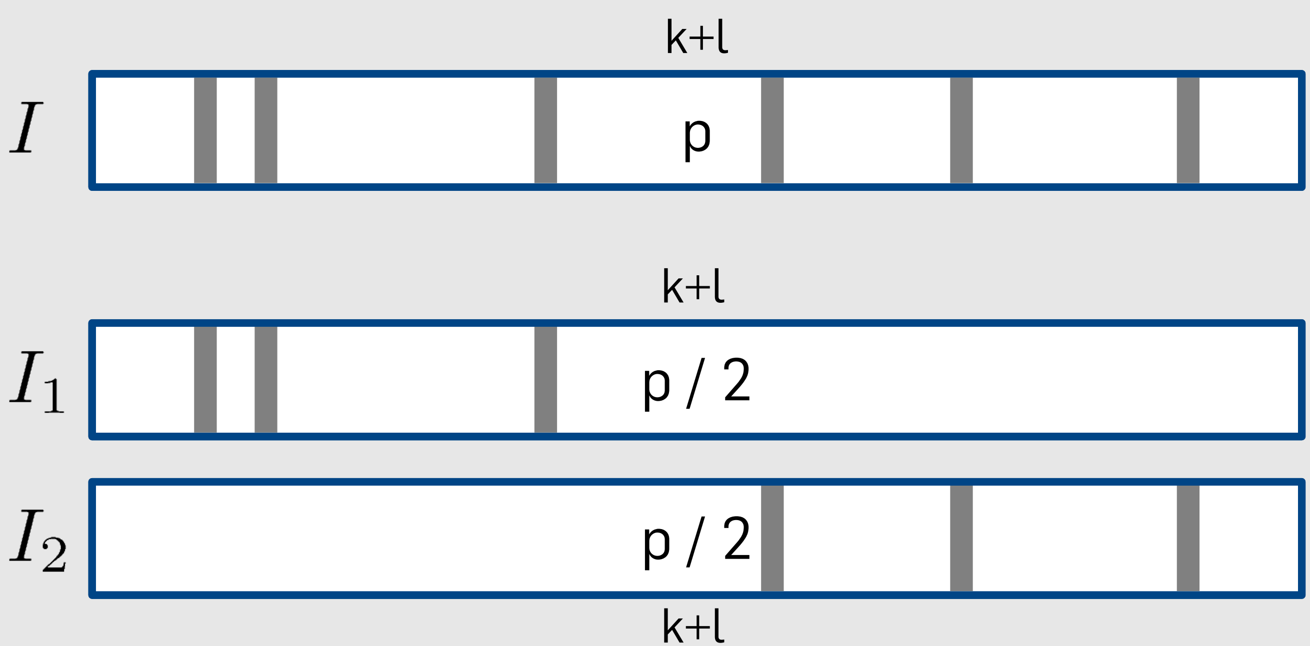
- Basic representation technique
- Arbitrary disjoint partition



Using Representations [MMT11]

Find a selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

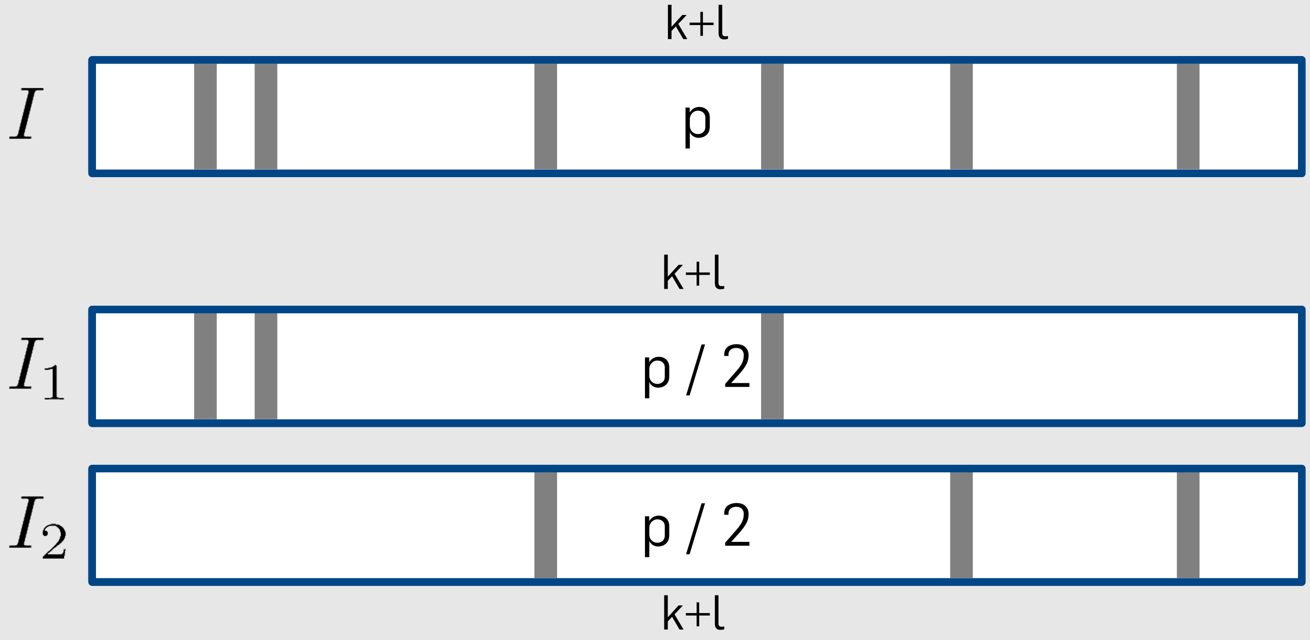
- Basic representation technique
- Arbitrary disjoint partition



Using Representations [MMT11]

Find a selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

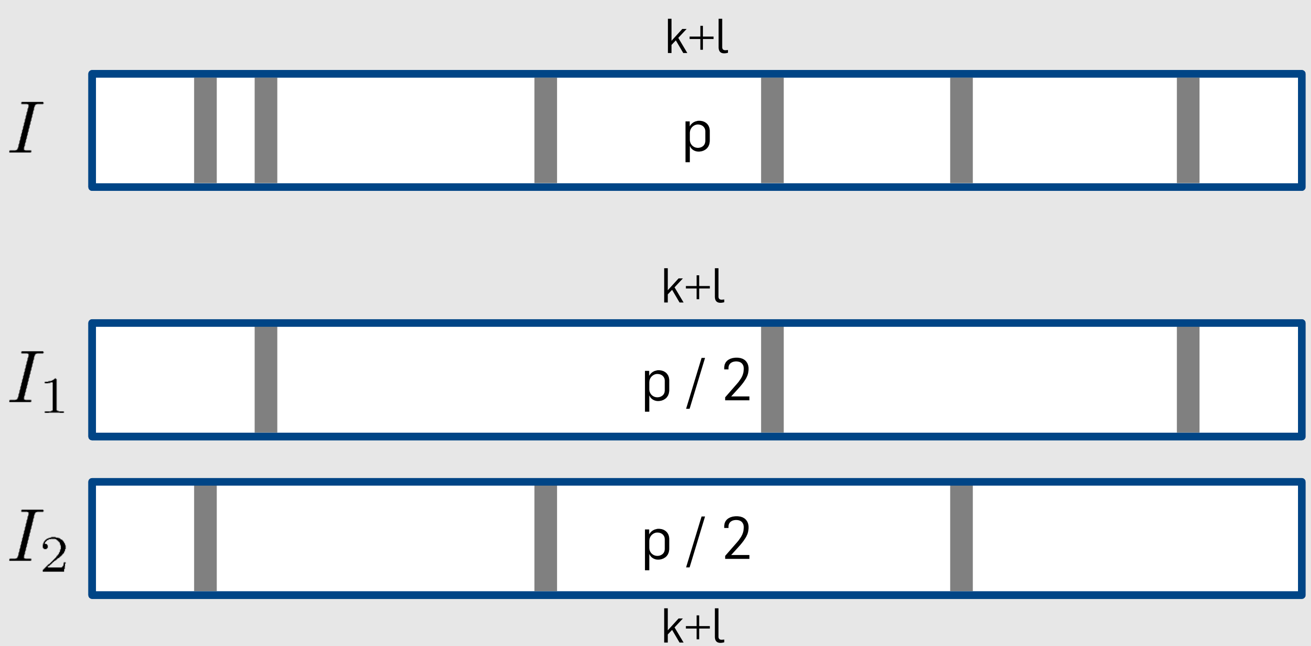
- Basic representation technique
- Arbitrary disjoint partition



Using Representations [MMT11]

Find a selection $I \subset [1, \ldots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

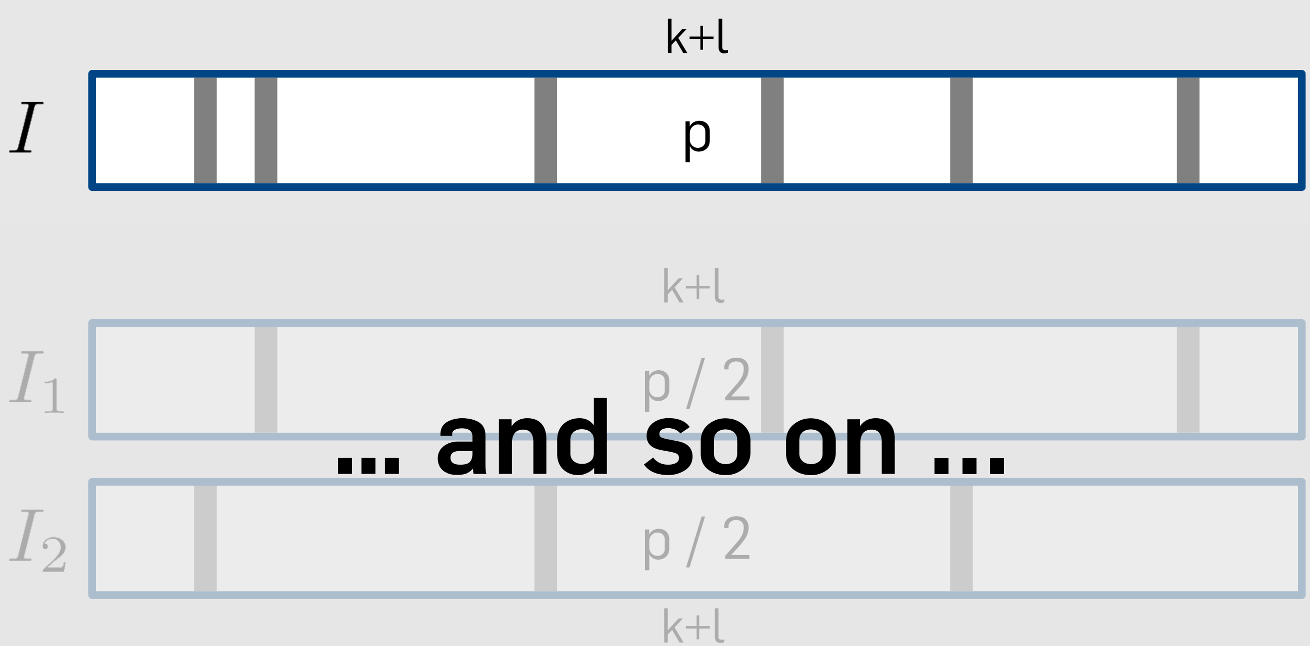
- Basic representation technique
- Arbitrary disjoint partition



Using Representations [MMT11]

Find a selection $I \subset [1, \dots, k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

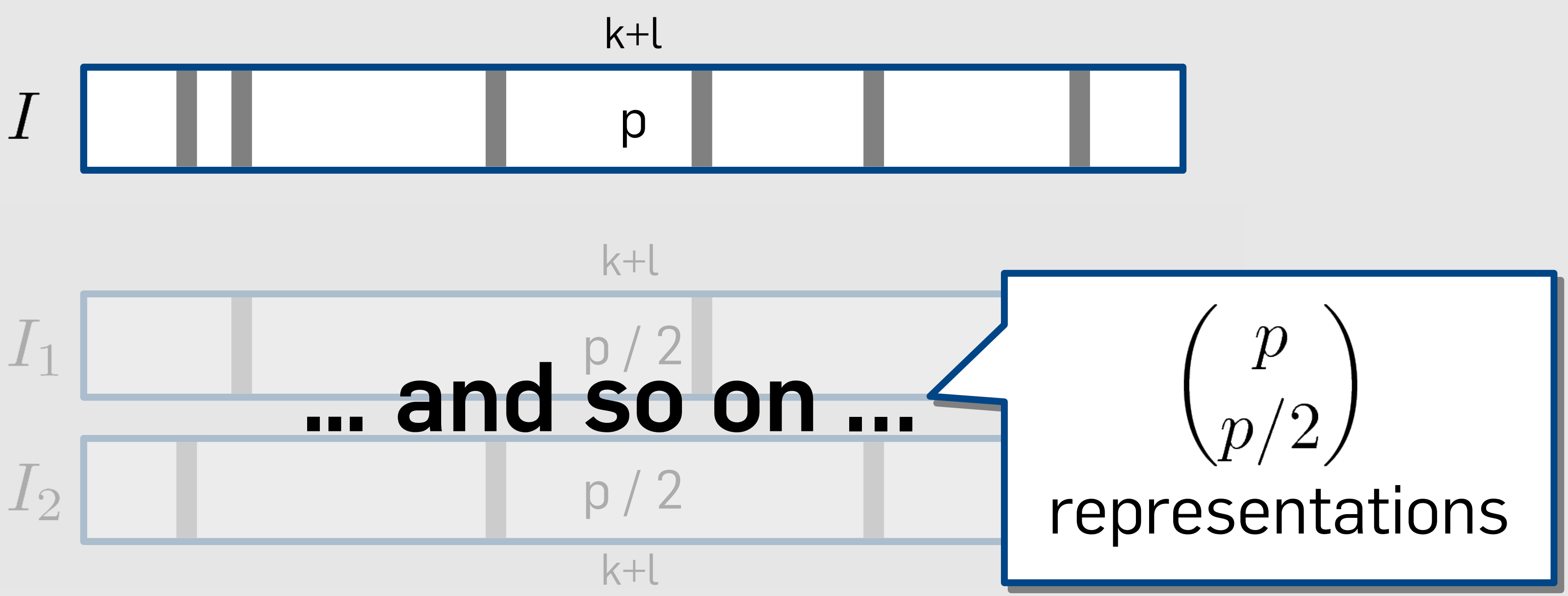
- Basic representation technique
- Arbitrary disjoint partition



Using Representations [MMT11]

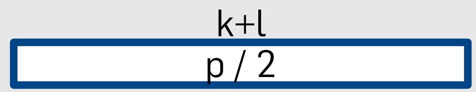
Find a selection $I \subset [1,\ldots,k+l], |I| = p$ with $\sum_{i \in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

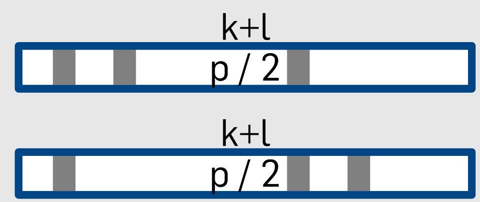
- Basic representation technique
- Arbitrary disjoint partition



Using Representations [MMT11]

Find a selection $I \subset [1,\ldots,k+l], |I| = p$ with $\sum_{i\in I} q_i = \begin{pmatrix} s_1 \\ \vdots \\ s_l \end{pmatrix}$

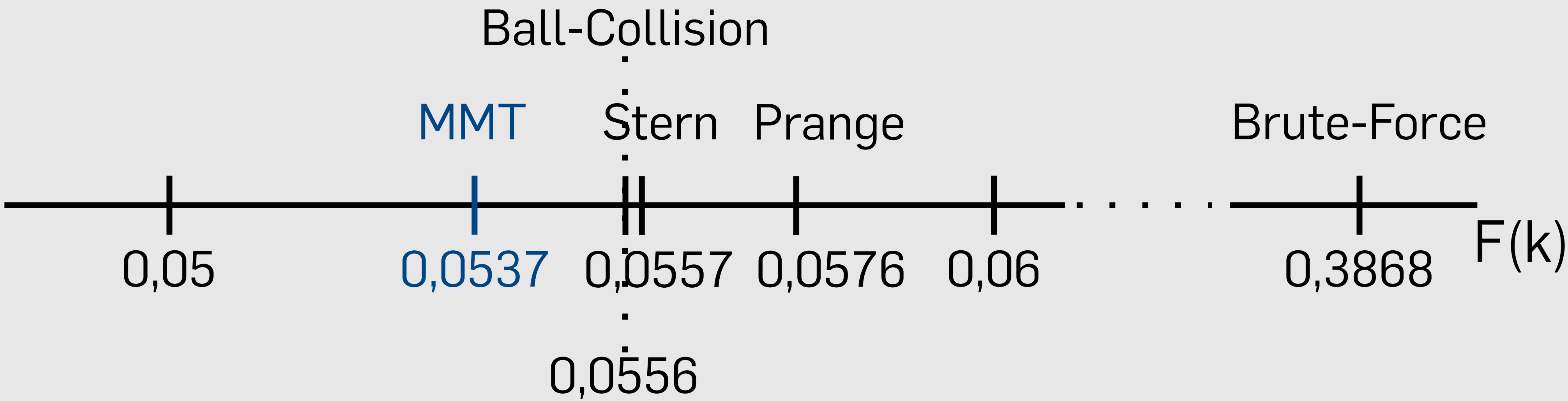
- Haystack = set of all 

- Needles = $\binom{p}{p/2}$ representations 

, , ...

- Bottleneck: Efficient computation of a $\frac{1}{\binom{p}{p/2}}$ - fraction of the haystack

Complexity Coefficients (BDD)



The Representation Technique

Optimizing the Representation Technique [BCJ11]

- r = number of needles
- |H'| = size of expanded haystack
- Ratio |H'| / r determines efficiency

→ **Increase r while keeping |H'| small**

The Representation Technique

Optimizing the Representation Technique [BCJ11]

- r = number of needles
- |H'| = size of expanded haystack
- Ratio |H'| / r determines efficiency

→ **Increase r while keeping |H'| small**

Can we use 1+1 = 0 to increase r ?

Using $1 + 1 = 0$

''Decoding Random Binary Linear Codes in $2^{n/20}$: How $1 + 1 = 0$ Improves Information Set Decoding.''

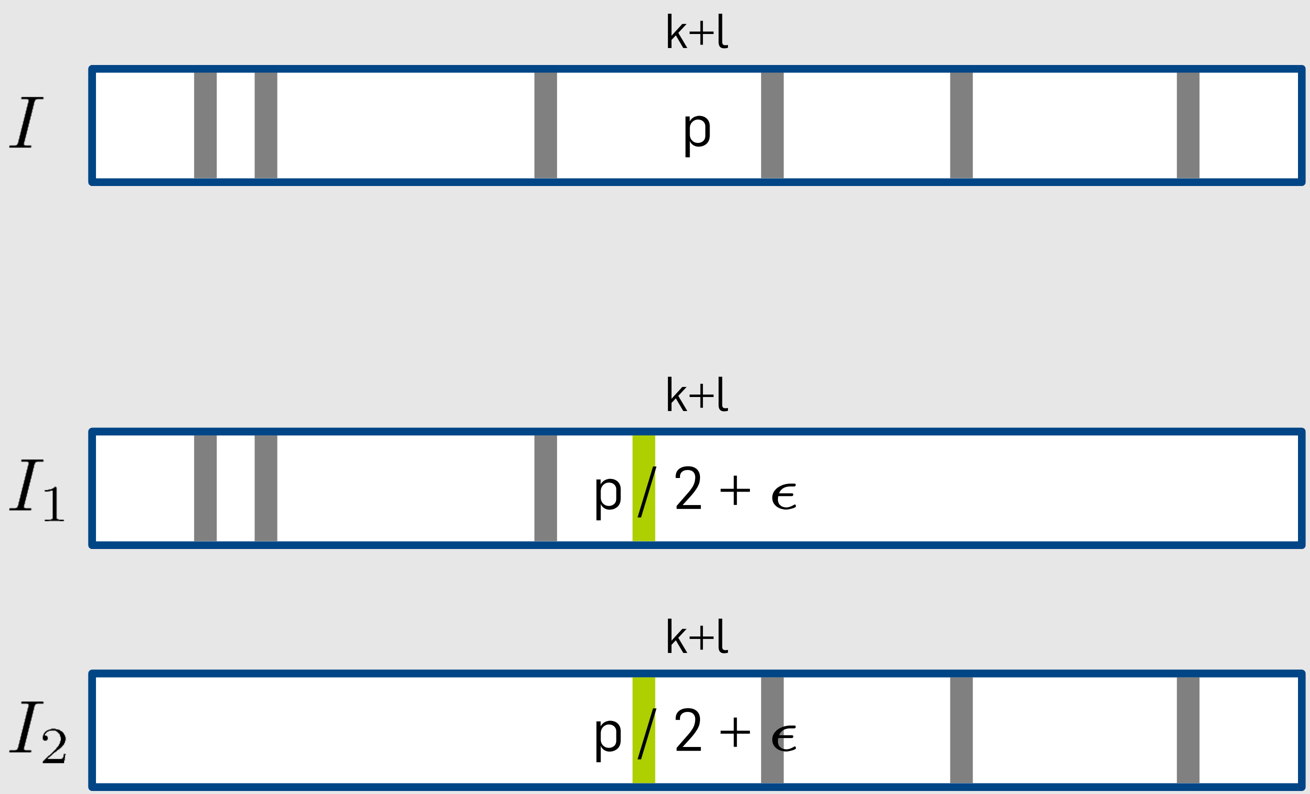
joint work with A.Becker, A.Joux & A.May (EUROCRYPT'12)

How to use $1 + 1 = 0$

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$

How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$

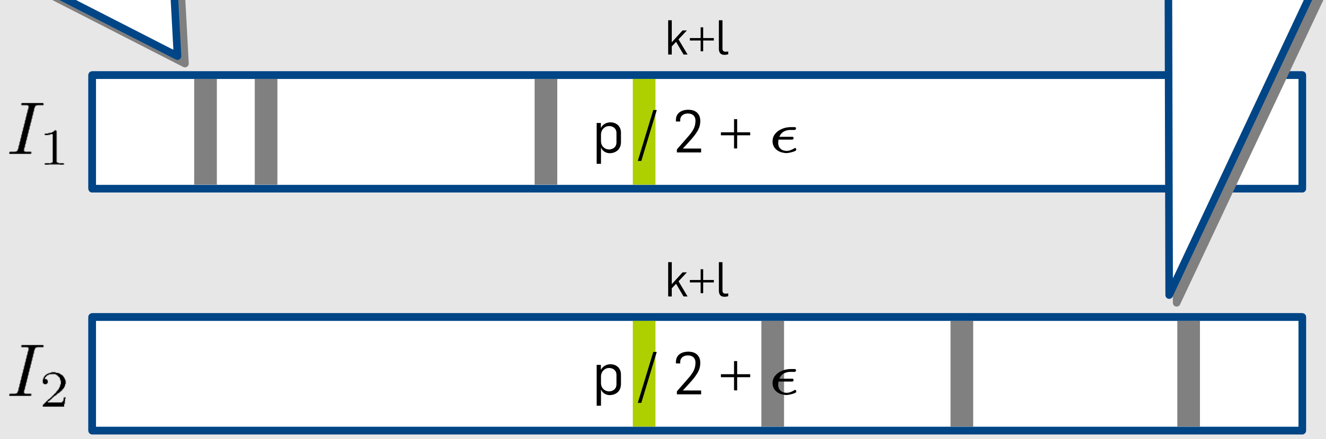


How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$

$$\sum_{i \in I_1} q_i$$

$$= \sum_{j \in I_2} q_j + s$$

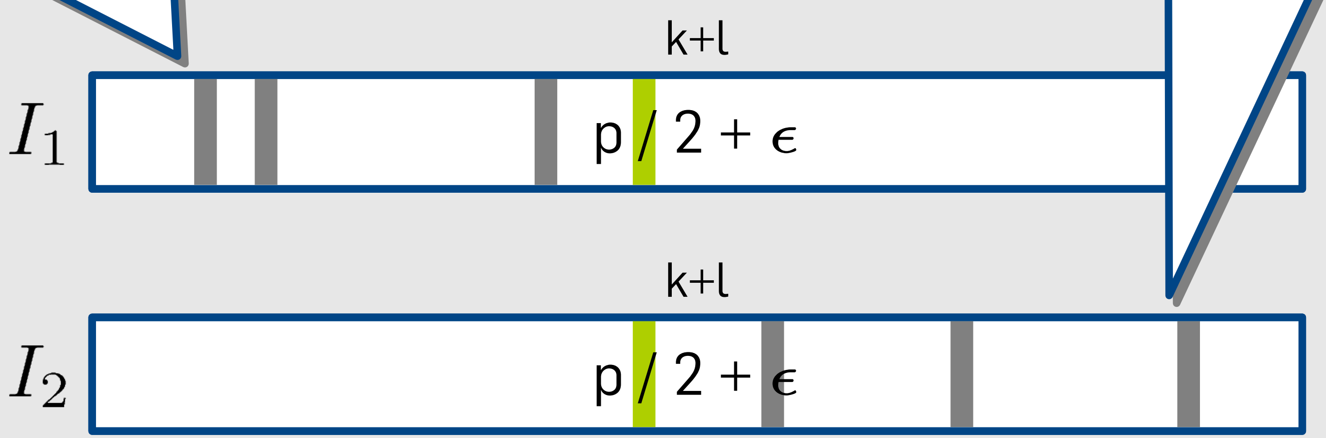


How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$

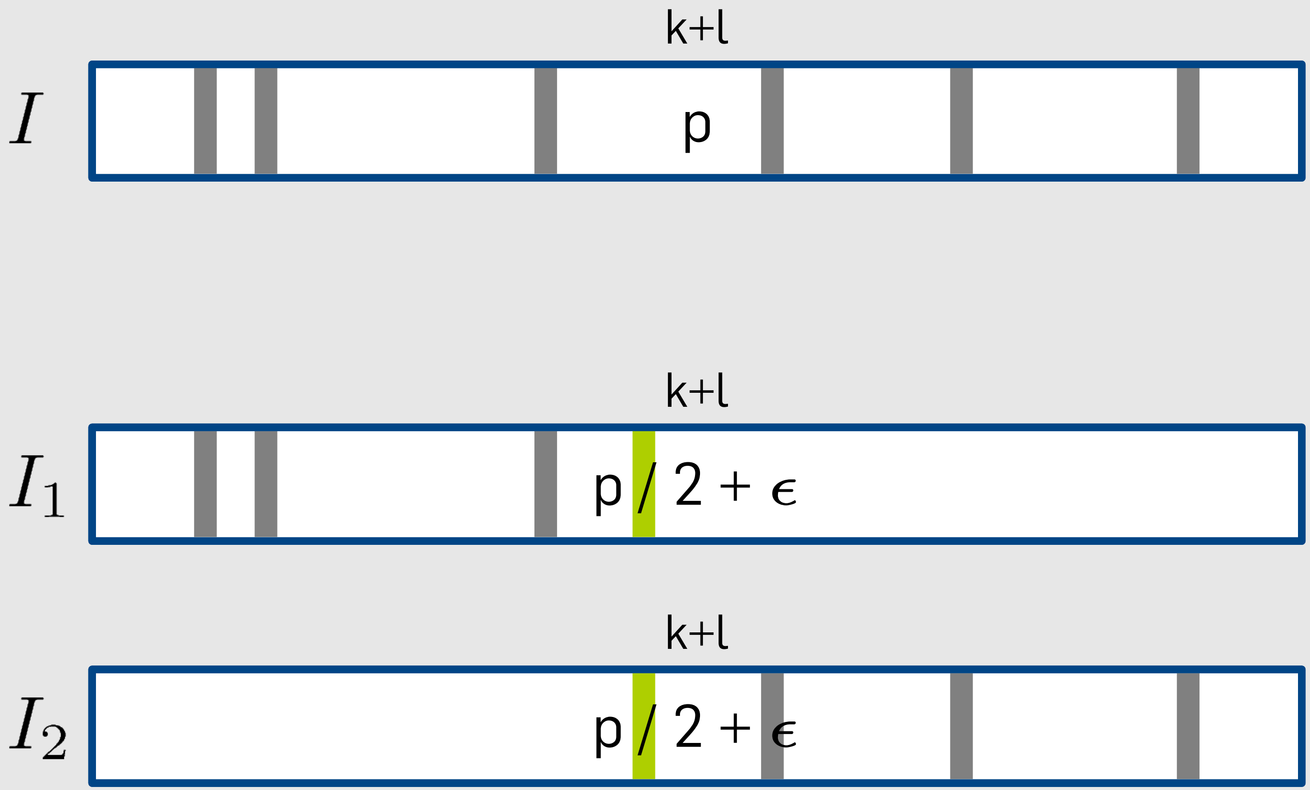
$$\sum_{i \in I_1} q_i \qquad = \sum_{j \in I_2} q_j + s$$

Double columns cancel out due to 1+1=0 !



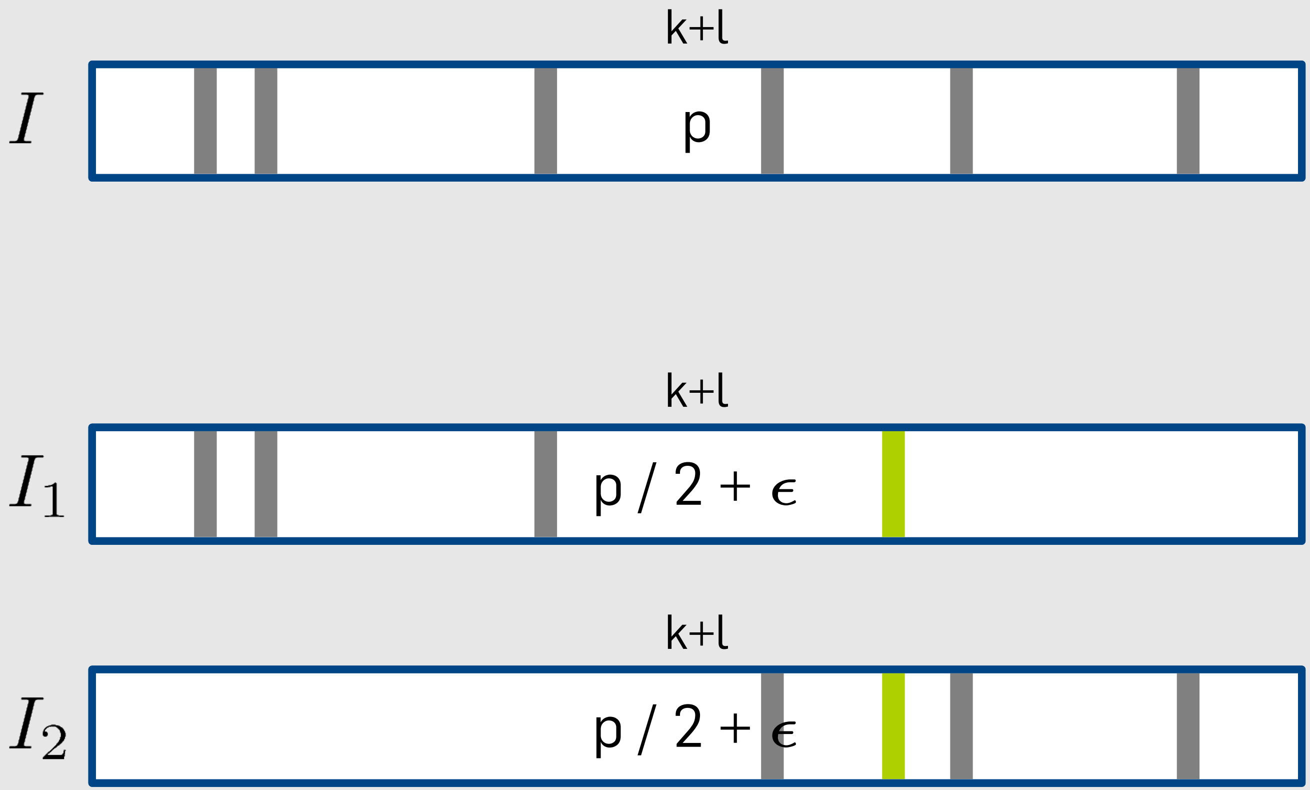
How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$



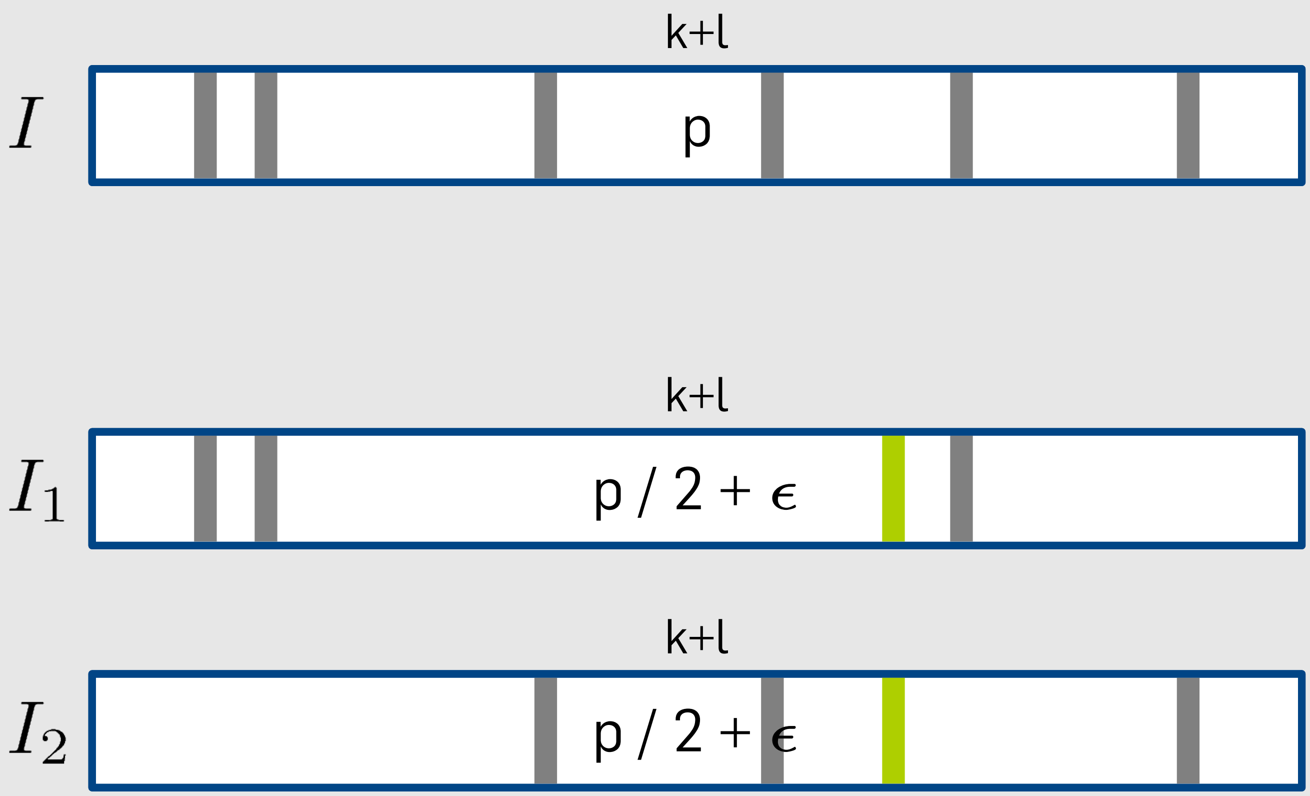
How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$



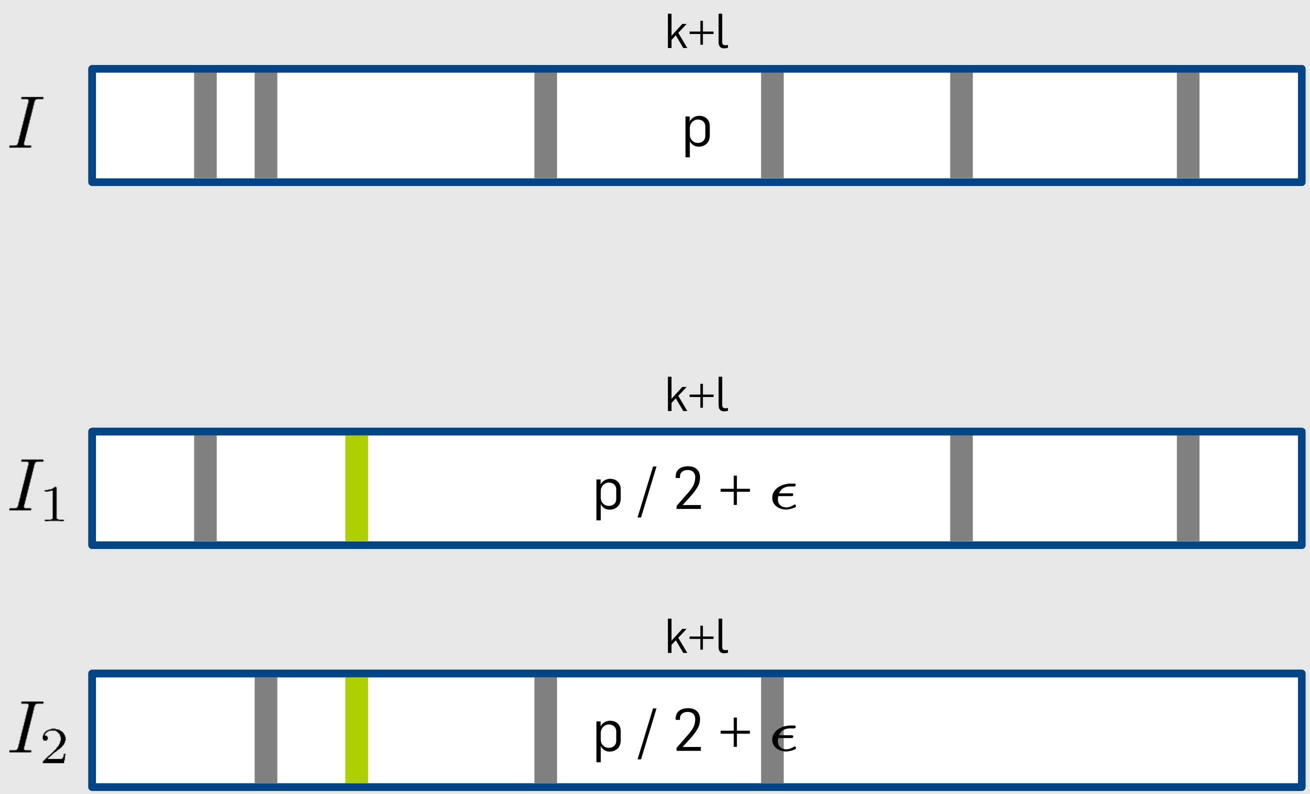
How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$



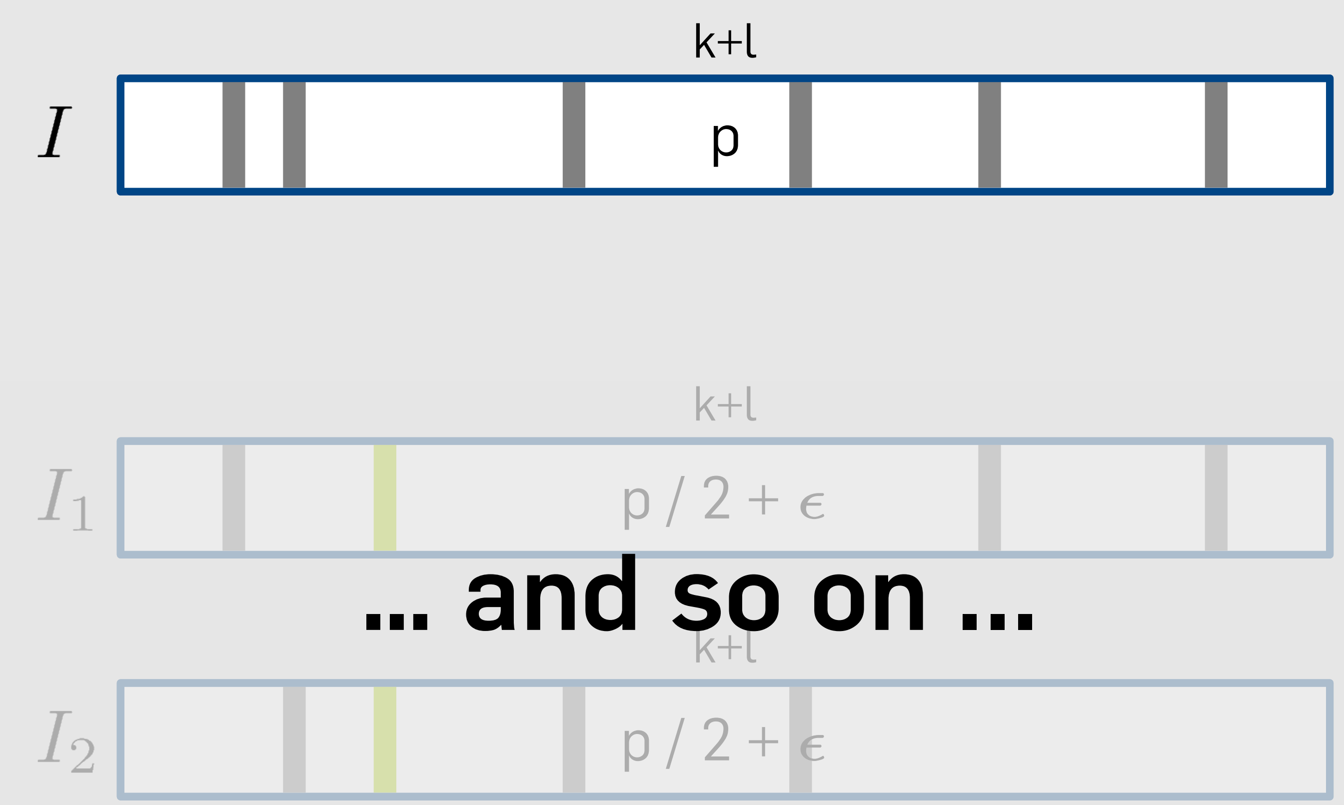
How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$



How to use 1 + 1 = 0

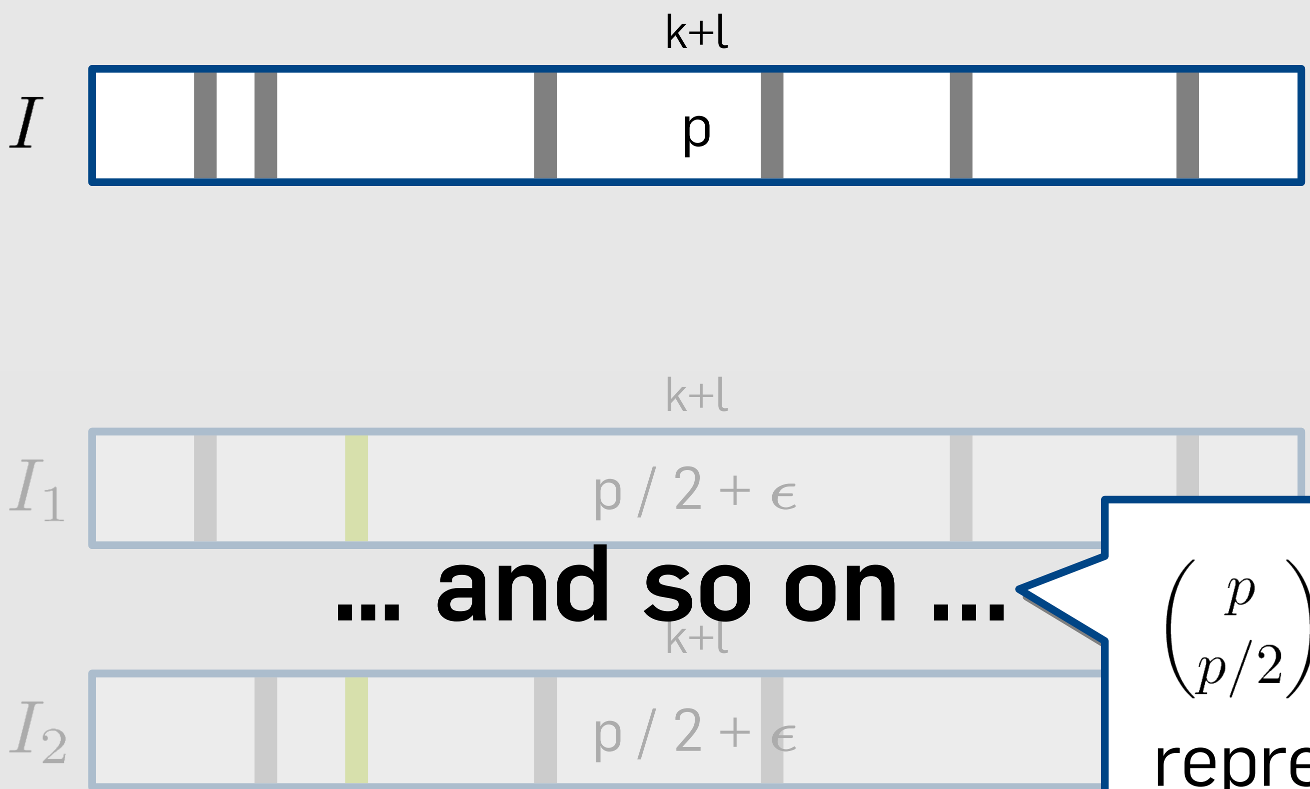
Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$



… and so on …

How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$



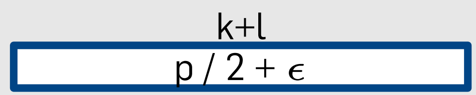
... and so on ...

$$\binom{p}{p/2} \cdot \binom{k+l-p}{\varepsilon}$$

representations

How to use 1 + 1 = 0

Write $I = I_1 \Delta I_2 := (I_1 \cup I_2) \setminus (I_1 \cap I_2)$ as the symmetric difference of intersecting sets $|I_1 \cap I_2| = \varepsilon$

- Haystack = set of all 

- Needles = $\underbrace{\binom{p}{p/2}\binom{k+l-p}{\varepsilon}}_{=:R}$ representations

, , ...

How can we compute a 1/R – fraction of the haystack ?

How to use 1 + 1 = 0

How can we compute a 1/R – fraction of the haystack ?

- Want to find *one* needle I_1 (and suitable I_2) with

$$\sum_{i \in I_1} q_i = \sum_{j \in I_2} q_j + s$$

$$\mathbf{q}_1 + \mathbf{q}_3 + \mathbf{q}_4 + \mathbf{q}_{11} = \mathbf{q}_2 + \mathbf{q}_4 + \mathbf{q}_7 + \mathbf{q}_{12} + \mathbf{s}$$

How to use 1 + 1 = 0

How can we compute a 1/R – fraction of the haystack ?

Uniform 0/1 - coordinates

he needle I_1 (and suitable I_2) with

$$\sum_{i \in I_1} q_i = \sum_{j \in I_2} q_j + s$$

$$\mathbf{q}_1 + \mathbf{q}_3 + \mathbf{q}_4 + \mathbf{q}_{11} = \mathbf{q}_2 + \mathbf{q}_4 + \mathbf{q}_7 + \mathbf{q}_{12} + \mathbf{s}$$

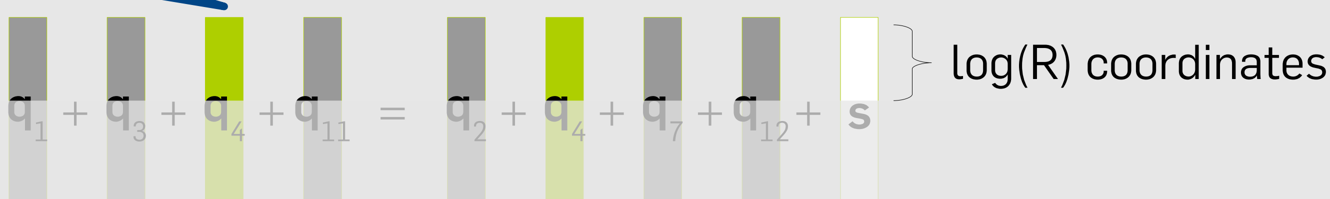
How to use 1 + 1 = 0

How can we compute a 1/R – fraction of the haystack ?

Uniform 0/1 - coordinates

ne needle I_1 (and suitable I_2) with

$$\sum_{i \in I_1} q_i = \sum_{j \in I_2} q_j + s$$



- Fix $\sum_{i \in I_1} q_i$ to **r** and $\sum_{j \in I_2} q_j$ to **s+r** on log(R) coordinates

→ Expect one needle to fulfill the extra constraint!

How to use 1 + 1 = 0

How can we compute a

ne need

Uniform 0/1 - coordinates

$\sum_{i\in I_1} q_i$

$\mathbf{q}_1 + \mathbf{q}_3 + \mathbf{q}_4 + \mathbf{q}_{11} = \mathbf{q}_2 +$ … $\mathbf{q}_{12} + \mathbf{s}$

log(R) coordinates

But how do we compute those restricted I_1 and I_2 ?

- Fix $\sum_{i\in I_1} q_i$ to **r** and $\sum_{j\in I_2} q_j$ to **s+r** on log(R) coordinates

→ **Expect one needle to fulfill the extra constraint!**

How to Fix log(R) Coordinates

- We want to compute

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1, \ldots, k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

How to Fix log(R) Coordinates

- We want to compute

On log(R) coordinates!

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1, \ldots, k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

How to Fix log(R) Coordinates

RUB

- We want to compute

On log(R) coordinates!

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1,\ldots,k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

- Choose random partition $\{1,\ldots,k+l\} = P_1 \dot\cup P_2$ with $|P_1| = |P_2| = \dfrac{k+l}{2}$

How to Fix log(R) Coordinates

RUB

- We want to compute

On log(R) coordinates!

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1,\ldots,k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

- Choose random partition $\{1,\ldots,k+l\} = P_1 \dot\cup P_2$ with $|P_1| = |P_2| = \frac{k+l}{2}$

- Compute base lists

$$\mathcal{B}_1 := \left\{ (J_1, \sum_{J_1} q_j) : |J_1| = \frac{p}{4} + \frac{\varepsilon}{2} \text{ and } J_1 \subset P_1 \right\}$$

$$\mathcal{B}_2 := \left\{ (J_2, \sum_{J_2} q_j + r) : |J_2| = \frac{p}{4} + \frac{\varepsilon}{2} \text{ and } J_2 \subset P_2 \right\}$$

How to Fix log(R) Coordinates

RUB

- We want to compute

On log(R) coordinates!

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1,\dots,k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

Merge $\mathcal{B}_1$ and $\mathcal{B}_2$ into $\mathcal{L}_1$!

- Compute base lists

$$\mathcal{B}_1 := \left\{ (J_1, \sum_{J_1} q_j) : |J_1| = \frac{p}{4} + \frac{\varepsilon}{2} \text{ and } J_1 \subset P_1 \right\}$$

$$\mathcal{B}_2 := \left\{ (J_2, \sum_{J_2} q_j + r) : |J_2| = \frac{p}{4} + \frac{\varepsilon}{2} \text{ and } J_2 \subset P_2 \right\}$$

How to Fix log(R) Coordinates

RUB

- We want to compute

On log(R) coordinates!

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1, \ldots, k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

Can be improved! Use representations again!

- Compute base lists

$$\mathcal{B}_1 := \left\{ (J_1, \sum_{J_1} q_j) : |J_1| = \frac{p}{4} + \frac{\varepsilon}{2} \text{ and } J_1 \subset P_1 \right\}$$

$$\mathcal{B}_2 := \left\{ (J_2, \sum_{J_2} q_j + r) : |J_2| = \frac{p}{4} + \frac{\varepsilon}{2} \text{ and } J_2 \subset P_2 \right\}$$

How to Fix log(R) Coordinates

RUB

- We want to compute

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1, \ldots, k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

where $p_1 = \frac{p}{2} + \varepsilon_1$

How to Fix log(R) Coordinates

RUB

- We want to compute

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1,\ldots,k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

where $p_1 = \frac{p}{2} + \varepsilon_1$

- Write $I_1 = J_1 \Delta J_2$ with $J_i \subset \{1,\ldots,k+l\}$

and $|J_i| = \frac{p_1}{2} + \varepsilon_2$

How to Fix log(R) Coordinates

- We want to compute

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1, \ldots, k+l\} : |I_1| = \frac{p}{2} + \varepsilon \text{ and } \sum_{I_1} q_i = r \right\}$$

where $p_1 = \dfrac{p}{2} + \varepsilon_1$

- Write $I_1 = J_1 \Delta J_2$ with $J_i \subset \{1, \ldots, k+l\}$ and $|J_i| = \dfrac{p_1}{2} + \varepsilon_2$
- Introduces $R_2 = \binom{p_1}{p_1/2}\binom{k+l-p_1}{\varepsilon_2}$ reps for each I_1

How to Fix log(R) Coordinates

RUB

- We want to compute

$$\mathcal{L}_1 = \left\{ I_1 \subset \{1,\ldots,k+l\} : |I_1| \right.$$

where $p_1 = \frac{p}{2} + \varepsilon_1$

Compute two lists $\mathcal{L}_1^1, \mathcal{L}_1^2$ containing a $1/R_2$-fraction of those J_1, J_2 !

- Write $I_1 = J_1 \Delta J_2$ with $J_i \subset \{1,\ldots,k+l\}$ and $|J_i| = \frac{p_1}{2} + \varepsilon_2$
- Introduces $R_2 = \binom{p_1}{p_1/2}\binom{k+l-p_1}{\varepsilon_2}$ reps for each I_1

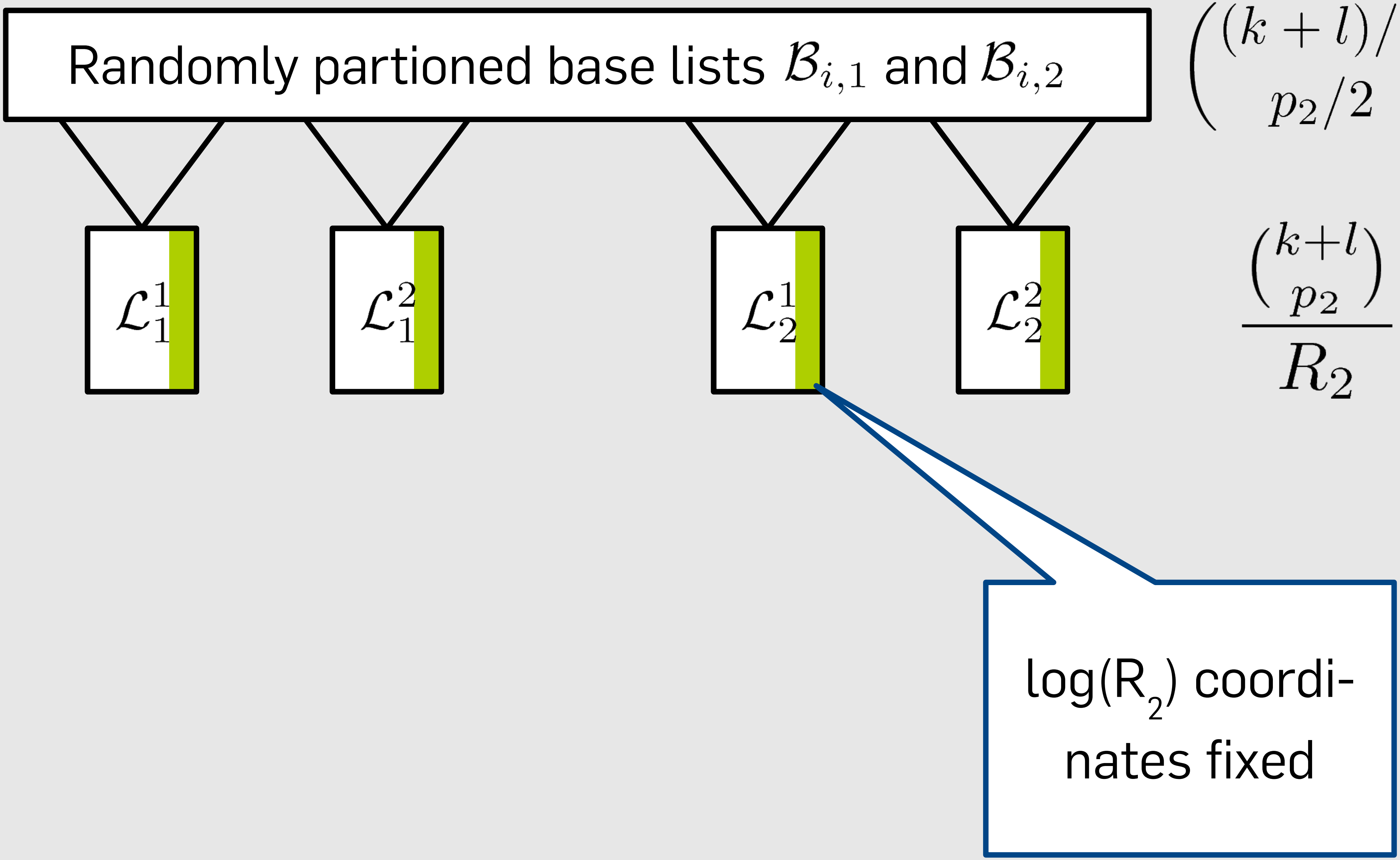
The Complete Computation Tree

RUB

Randomly partioned base lists $\mathcal{B}_{i,1}$ and $\mathcal{B}_{i,2}$

$$\binom{(k+l)/2}{p_2/2}$$

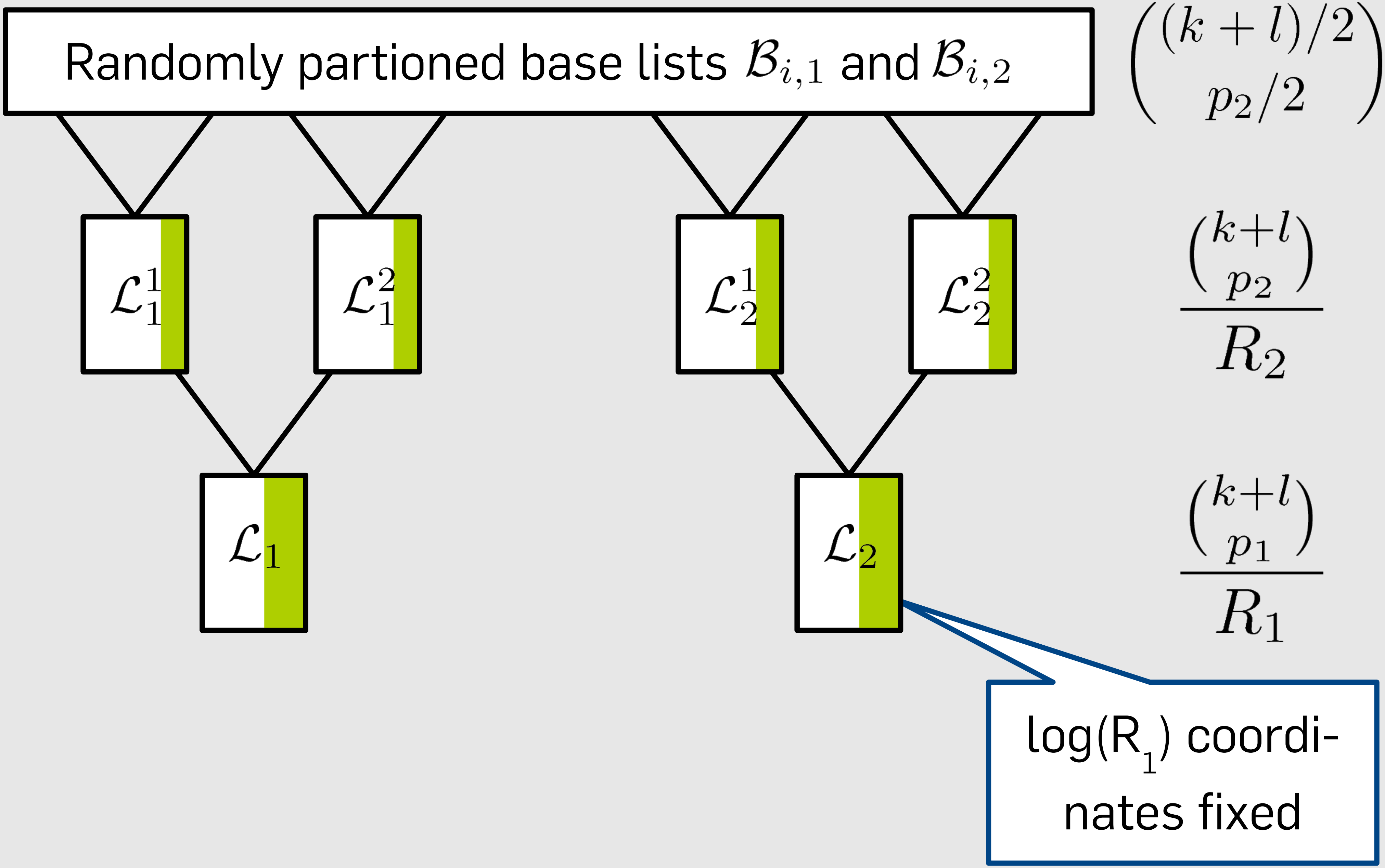
The Complete Computation Tree



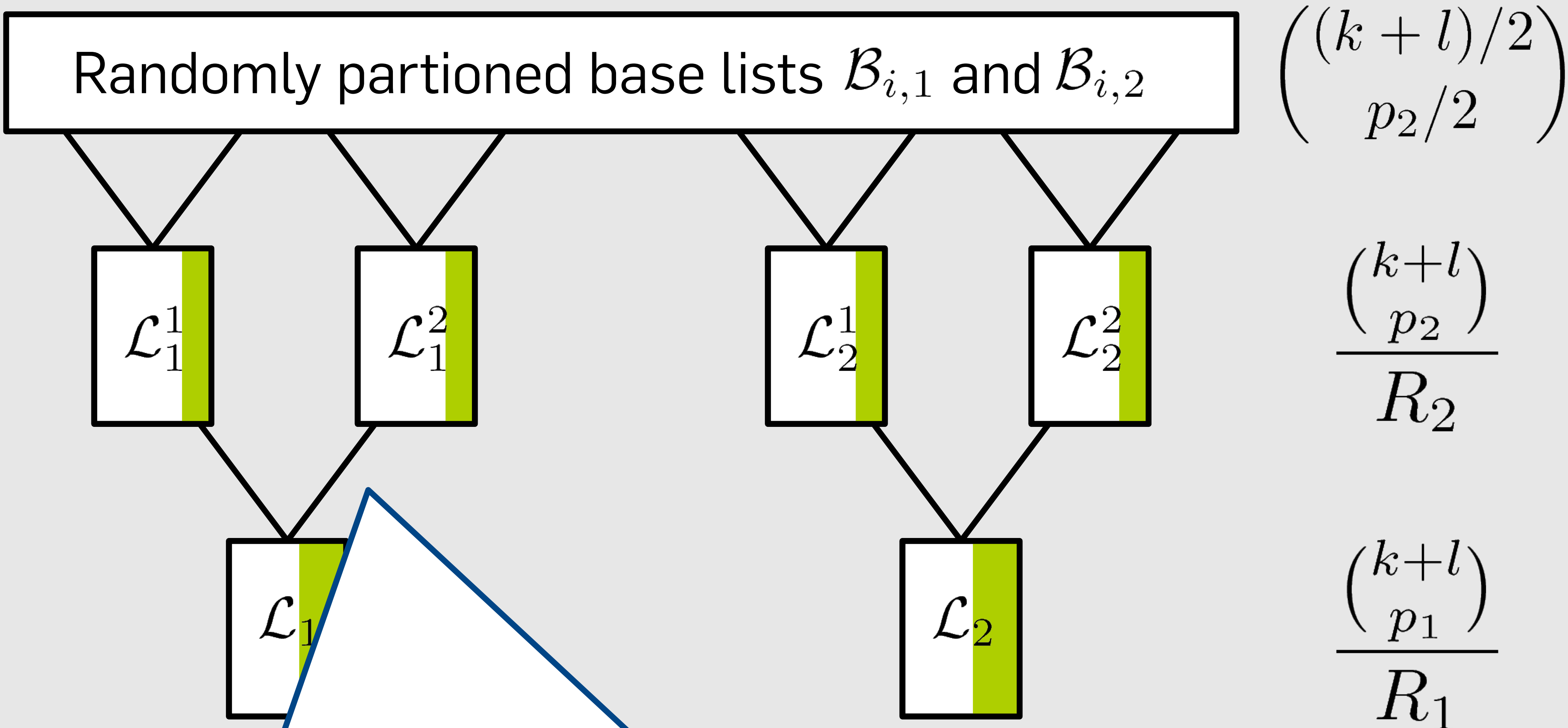
$$\binom{(k+l)/2}{p_2/2}$$

$$\frac{\binom{k+l}{p_2}}{R_2}$$

The Complete Computation Tree

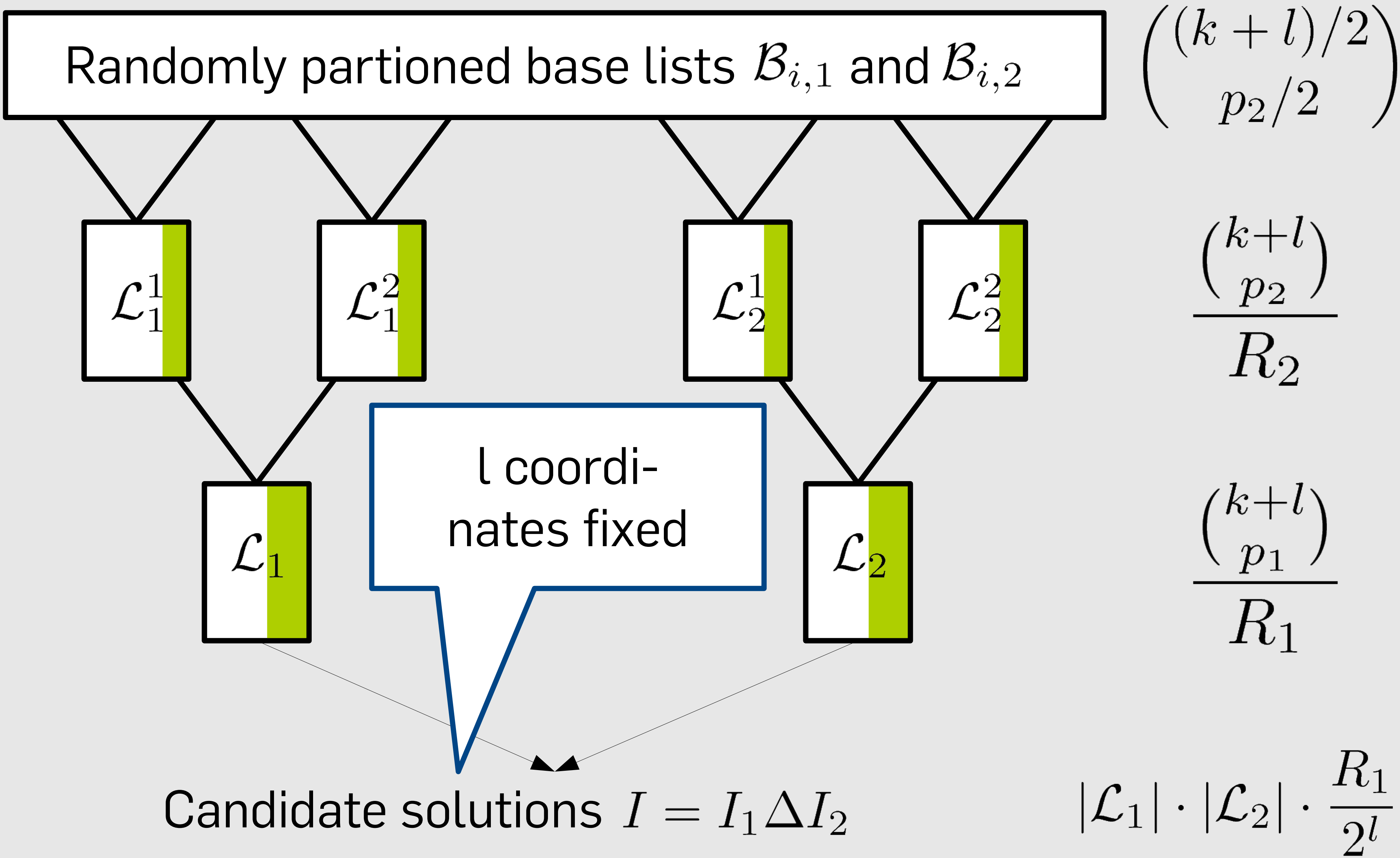


The Complete Computation Tree



Warning! Inconsistencies (i.e. matchings of false weight) have to be sorted out!

The Complete Computation Tree



Some Technicalities

- Need to exclude "badly distributed" $\mathbf{q}_1 , \ldots , \mathbf{q}_{k+l}$

  → intermediate lists become too large (abort)

  → solution get's lost w.h.p.

Some Technicalities

- Need to exclude "badly dist

  → intermediate lists bec

  → solution get's lost w.h.p.

- Method introduces extra inverse-polynomial failure probability (due to disjoint partitions on bottom level)

Can be avoided in implementations! Do non-disjoint base lists!

Some Technicalities

RUB

- Need to exclude "badly distributed“ $\mathbf{q}_1 , \ldots , \mathbf{q}_{k+l}$

  → intermediate lists become too large (abort)

  → solution get's lost w.h.p.

- Method introduces extra inverse-polynomial failure probability (due to disjoint partitions on bottom level)

- We only fix parameters to guarantee

$$E[\#\ \text{surviving reps}] \geq 1$$

Some Technicalities

- Need to exclude "badly distributed" $\mathbf{q}_1, \ldots, \mathbf{q}_{k+l}$

  → intermediate lists become too large (abort)

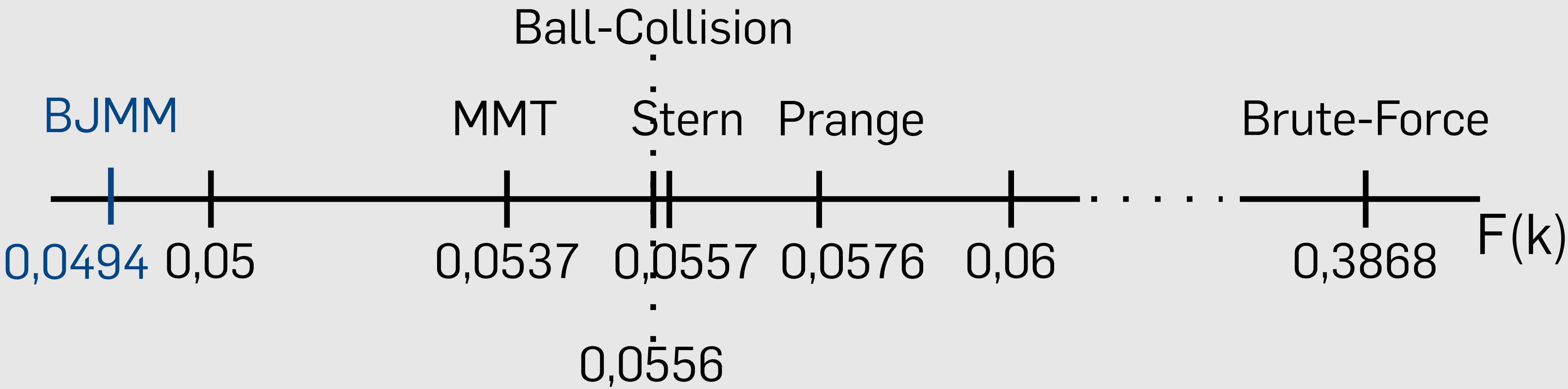
  → solution get's lost w.h.p.

- Method introduces extra inverse-polynomial failure probability (due to disjoint partitions on bottom level)

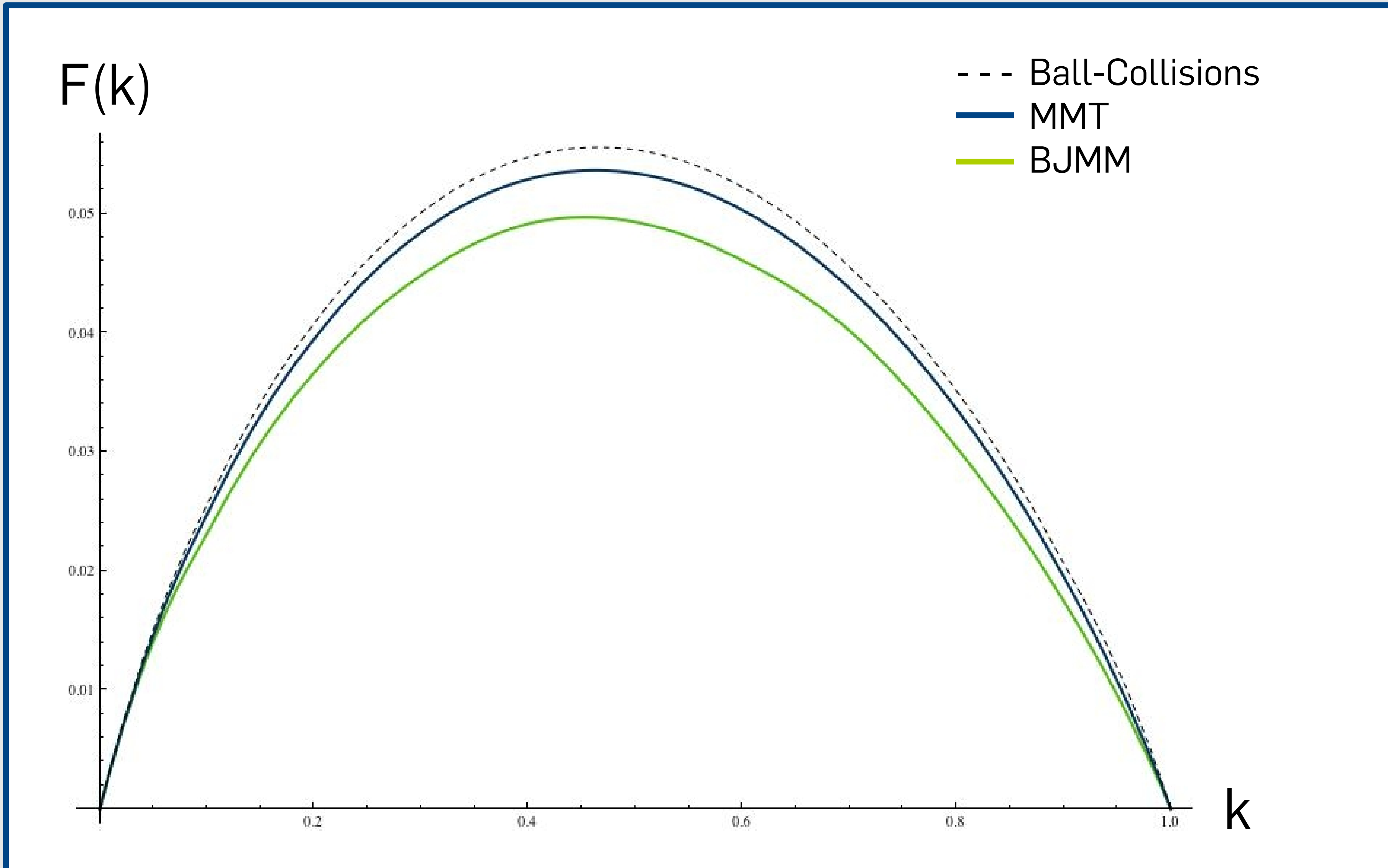
- We only fix parameters to guarantee

$$E[\#\text{ surviving reps}] \geq 1$$

Main Result F(k) ≤ 0.0494



Main Result $F(k) \leq 0.0494$



In Practical terms...

- 256-Bit security for McEliece revisited

  → $[n,k,d] = [6624,5129,117]$

- Exact complexity analysis (using tricks from [BLP08])

  → Stern $\approx 2^{256}$

  → Ball-Collisions $\approx 2^{254}$

  → Our Algorithm $\approx 2^{239}$

- Parameters: $l = 286$ $\quad p = 44$ $\quad \epsilon_1 = 12$ $\quad \epsilon_2 = 1$

In Practical terms...

- 256-Bit security for McEliec

  → [n,k,d] = [6624,5129,117]

- Exact complexity analysis (u

  → Stern ≈ 2^{256}

  → Ball-Collisions ≈ 2^{254}

  → Our Algorithm ≈ 2^{239}

- Parameters: $l = 286$ $p = 44$ $\epsilon_1 = 12$ $\epsilon_2 = 1$

Toolkit for optimal parameter choices will be available soon (includes all ISD algorithms up-to-date)

Wrapping up...

RUB

Summary

- Using 1+1=0 introduces extra representations
- Asymptotically fastest generic decoding algorithm
- Even practical impact (e.g. for high security levels of McEliece)
- Full Version ePrint 2012/026

Wrapping up...

RUB

Summary

- Using 1+1=0 introduces extra representations
- Asymptotically fastest generic decoding algorithm
- Even practical impact (e.g. for high security levels of McEliece)
- Full Version ePrint 2012/026

Thank you!