RUHR-UNIVERSITÄT BOCHUM

RUB

# Decoding Linear Codes with High Error Rate and its Impact for LPN Security

**PQCrypto 2018**, 09.-11.04.2018

**Leif Both**, Alexander May
Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
Faculty of Mathematics

hgi
Horst Görtz Institut
für IT-Sicherheit

# Our work

- Improved running times for decoding of random linear codes.

| | State of the art | Our algorithm |
|---|---|---|
| Full Distance (FD) | $2^{0.0953n}$ | $2^{0.0885n}$ |
| Half Distance (HD) | $2^{0.0473n}$ | $2^{0.0465n}$ |

- Based on the BJMM algorithm (Becker, Joux, May, Meurer EC2012) and Nearest Neighbors (May, Ozerov EC2015).
- Works best for high error rates.
- Application: Hybrid algorithm for LPN (Esser, Kübler, May Crypto2017).

# On Linear Codes

## Definition (Linear Code)

A *linear code* $C$ is a $k$-dimensional subspace of $\mathbb{F}_2^n$.

- Alternative definition via **Parity Check matrix** $P$

$$C = \{\mathbf{c} \in \mathbb{F}_2^n \mid P\mathbf{c} = \mathbf{0}\}, \text{ where } P \in \mathbb{F}_2^{(n-k)\times n}.$$

## Definition (Distance)

For a linear code $C$ the *distance* is defined as

$$d = \min_{\mathbf{c}\neq\mathbf{c}'\in C}\{\Delta(\mathbf{c}, \mathbf{c}')\}.$$

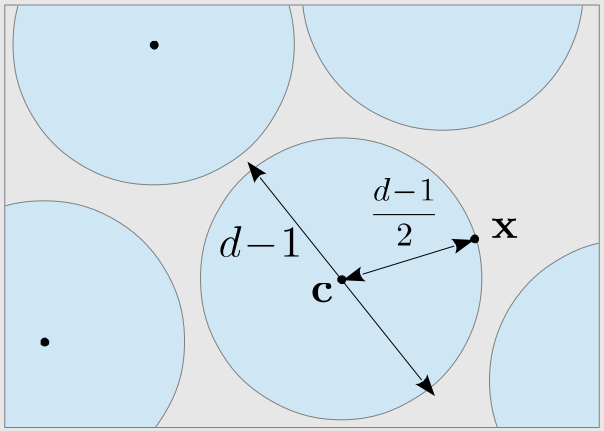
# The Decoding Problem



## Definition (Decoding Problem)

**Given:** $P, \omega, \mathbf{x} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in C$, $\Delta(\mathbf{e}) = \omega$
**Find:** $\mathbf{e}$ $(\Rightarrow \mathbf{c} = \mathbf{x} + \mathbf{e})$.

- Unique decoding of $\mathbf{x}$ if $\omega \leq \frac{d-1}{2}$.
- **HD Decoding**: $\omega = \frac{d-1}{2}$.
- **FD Decoding**: $\omega = d$.




## Definition (Syndrome)

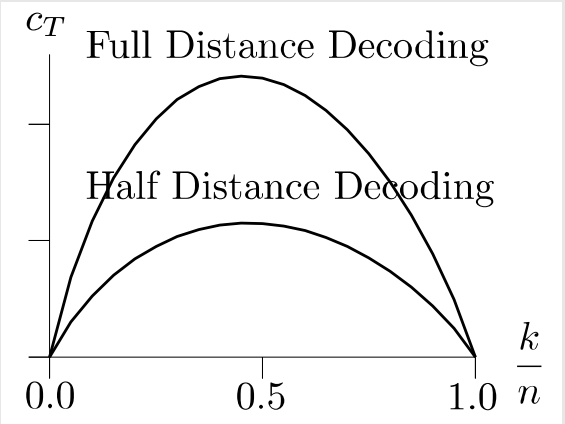
The *Syndrome* $\mathbf{s}$ of a vector $\mathbf{x}$ is defined as $\mathbf{s} := P\mathbf{x}$.

- $\mathbf{s} = P\mathbf{x} = P\mathbf{c} + P\mathbf{e} \quad \Leftrightarrow \quad \mathbf{s} = P\mathbf{e}$.

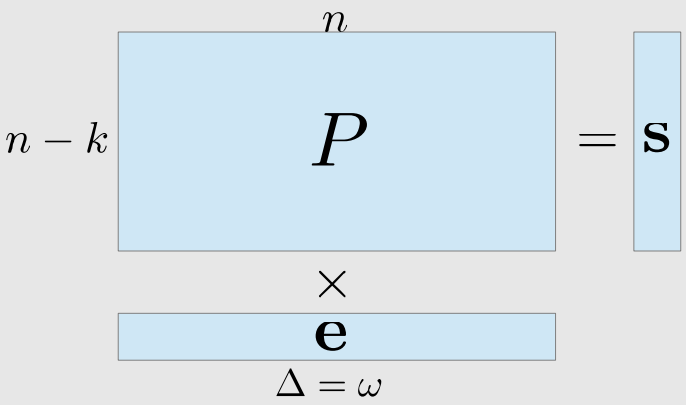
# Compare Decoding Algorithms



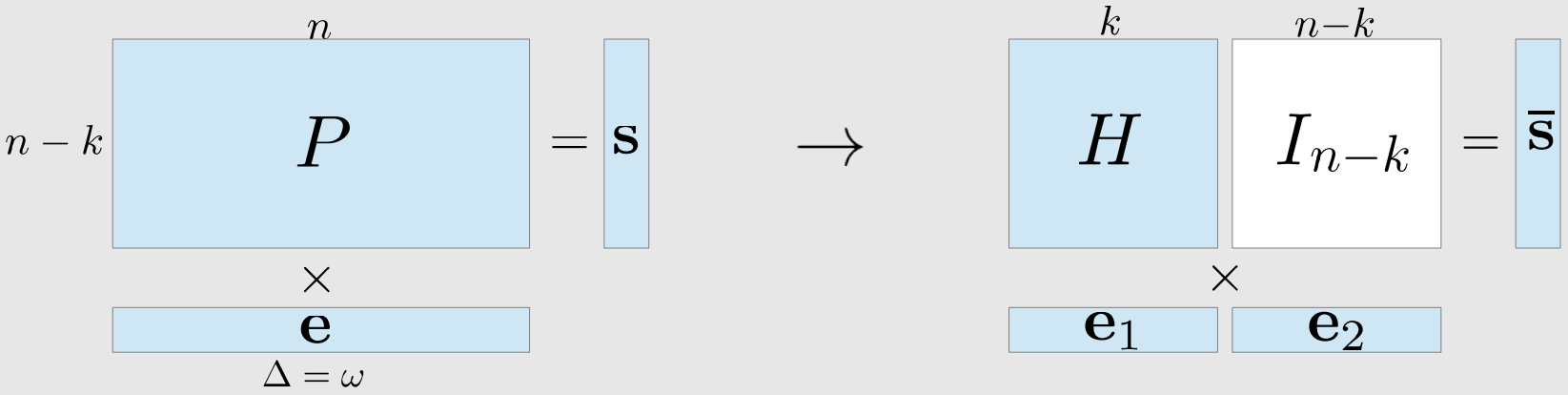
- Running Time $T(n, k, d)$.
- Use the Gilbert-Varshamov bound
  $\Rightarrow d = f(n, k) \Rightarrow T(n, k, d) = T(n, k)$.
- Worst case running time: $T(n) = \max_k\{T(n, k)\}$.
- Assumption: Exponential complexity of HD/FD decoding
  $\Rightarrow T(n) = 2^{c_T n}$.



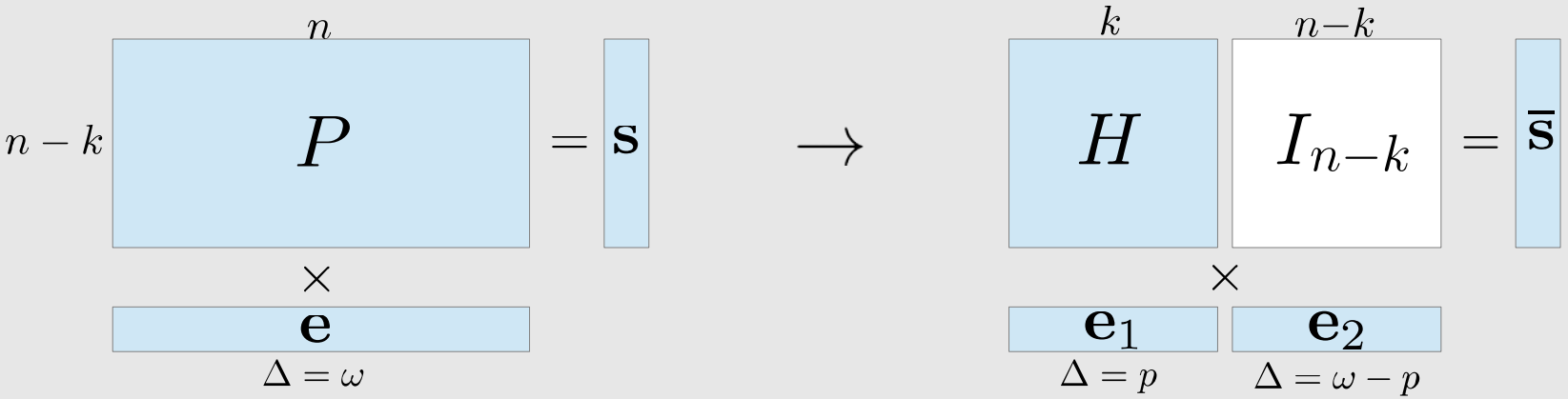
# Prange: Basic Idea for Decoding (1962)



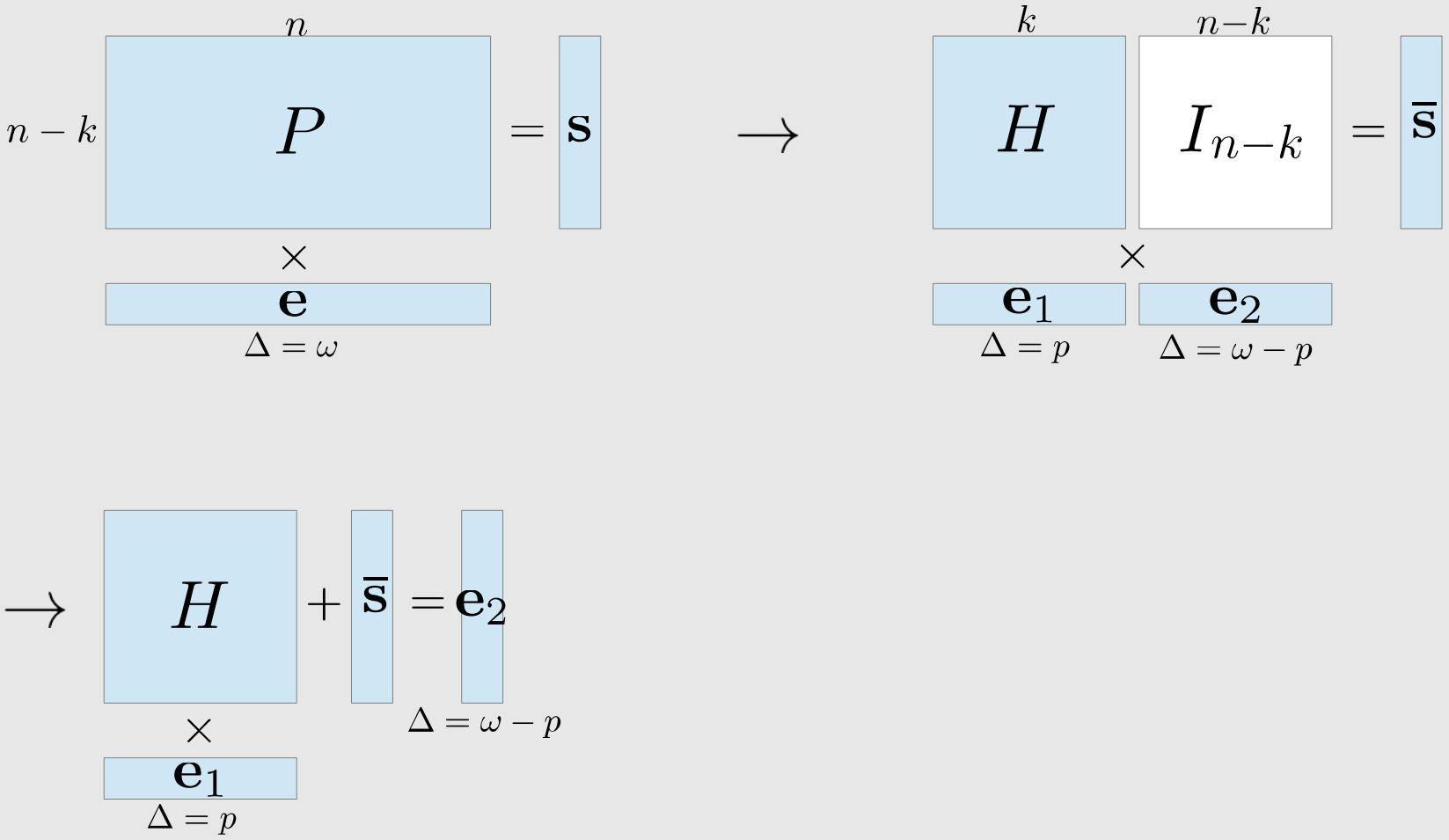
# Prange: Basic Idea for Decoding (1962)



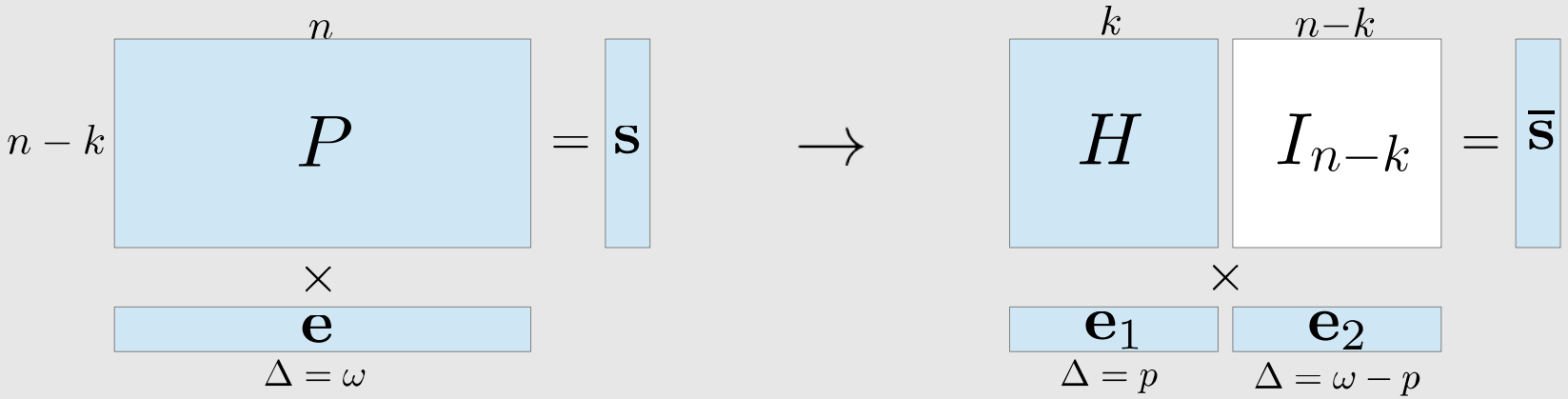
# Prange: Basic Idea for Decoding (1962)

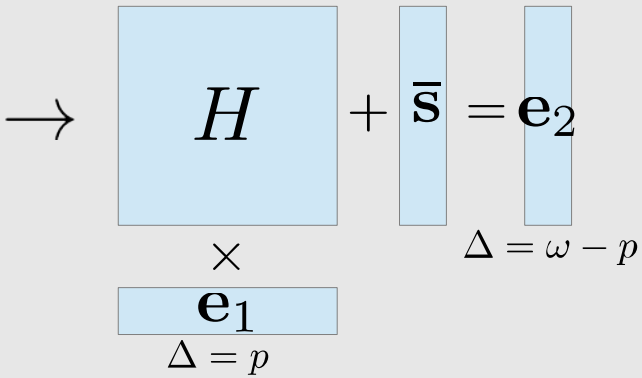


# Prange: Basic Idea for Decoding (1962)



# Prange: Basic Idea for Decoding (1962)



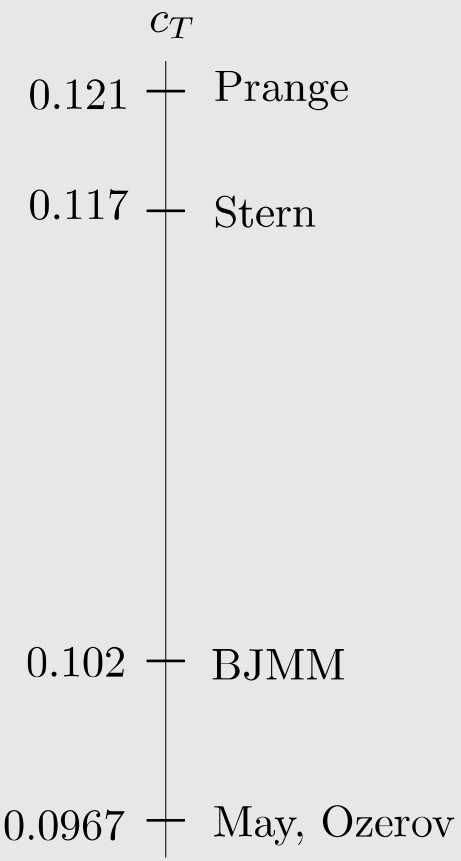




## Algorithm (Idea)

1. Bring $P$ into systematic form.
2. Permute columns.
3. Enumerate all $\mathbf{e}_1$.
4. Check if $\Delta(H\mathbf{e}_1 + \bar{\mathbf{s}}) = \omega - p$.

# Advanced Ideas

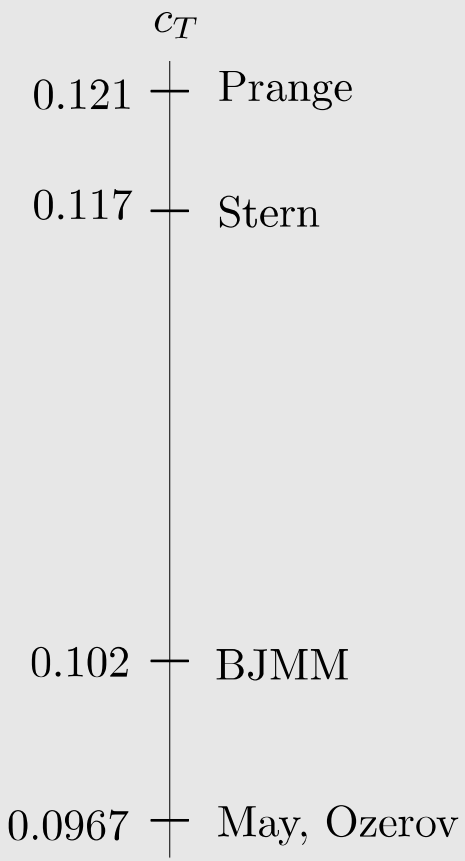
- Exact matching on some coordinates (Stern 1989).
- Meet in the middle (Stern 1989).
- Representations techniques (BJMM EC2012).
- Binary search tree (BJMM EC2012).
- Nearest Neighbors (May, Ozerov EC2015).



# Advanced Ideas (Our work)



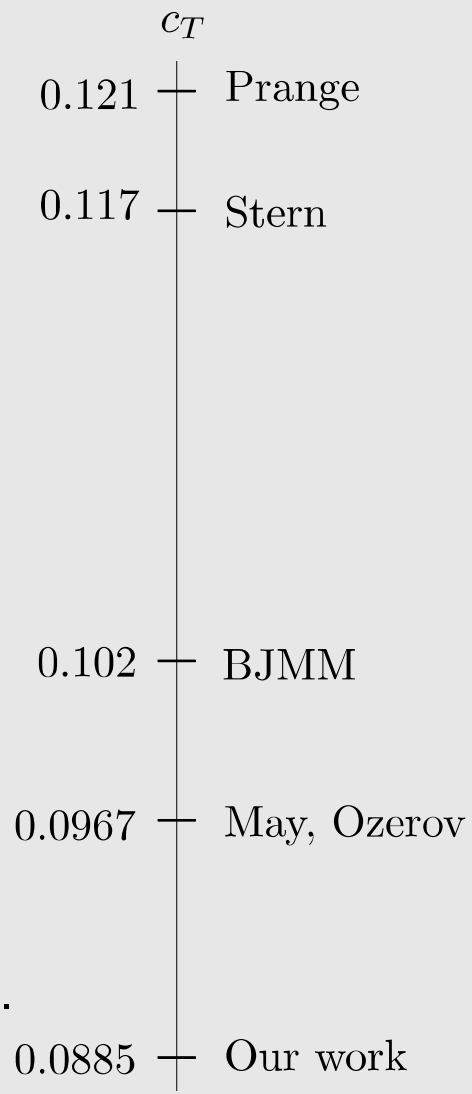
- No exact matching (Bernstein et al. Crypto2011).
- Meet in the middle (Stern 1989).
- Representations techniques (BJMM EC2012).
- Binary search tree (BJMM EC2012).
- Nearest Neighbors (May, Ozerov EC2015).



# Advanced Ideas (Our work)



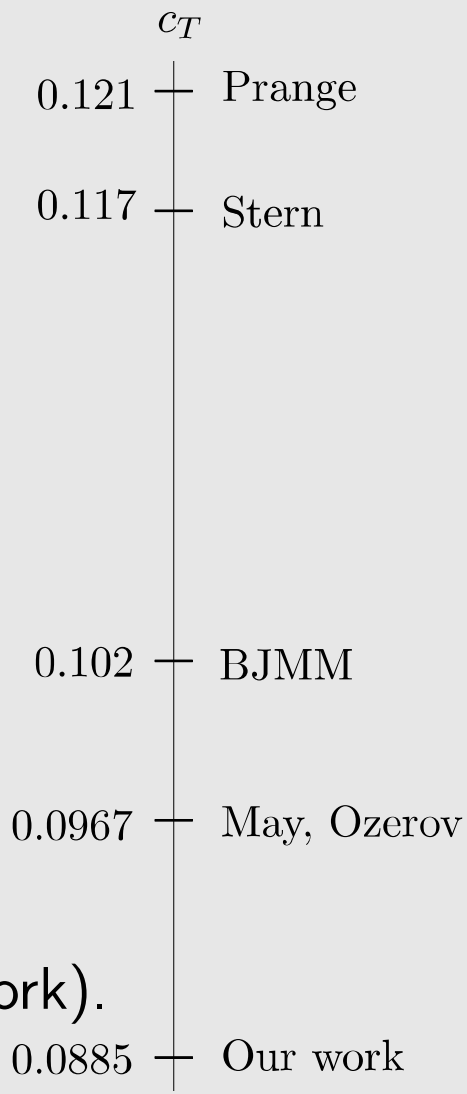
- No exact matching (Bernstein et al. Crypto2011).
- Meet in the middle (Stern 1989).
- Representations techniques (BJMM EC2012).
- Binary search tree (BJMM EC2012).
- Nearest Neighbors (May, Ozerov EC2015).
- Division into blocks of different weight (Our work).



# Advanced Ideas (Our work)

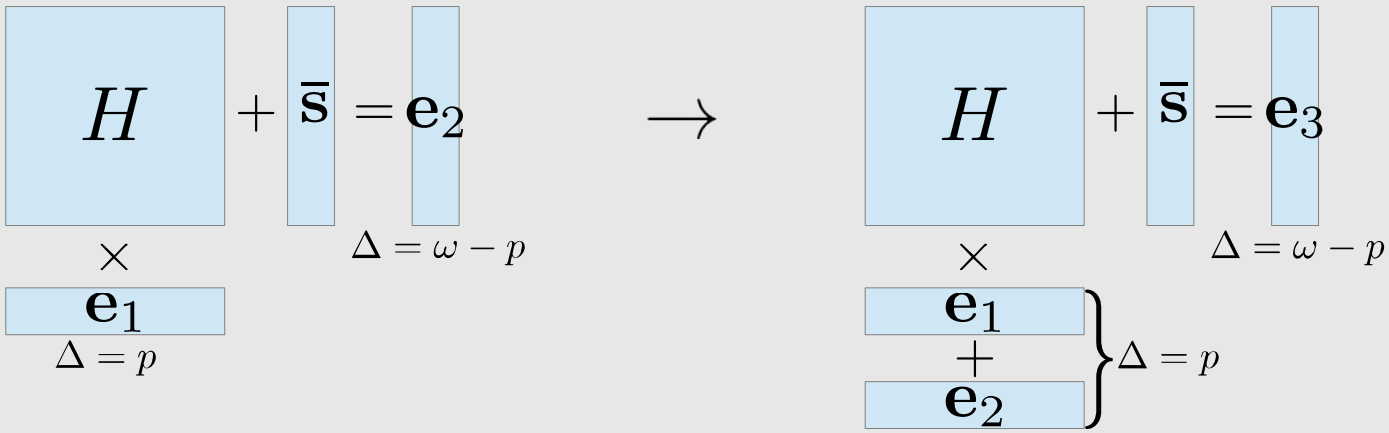


- No exact matching (Bernstein et al. Crypto2011).
- Meet in the middle (Stern 1989).
- **Representations techniques** (BJMM EC2012).
- Binary search tree (BJMM EC2012).
- Nearest Neighbors (May, Ozerov EC2015).
- **Division into blocks of different weight** (Our work).



# Representations Techniques

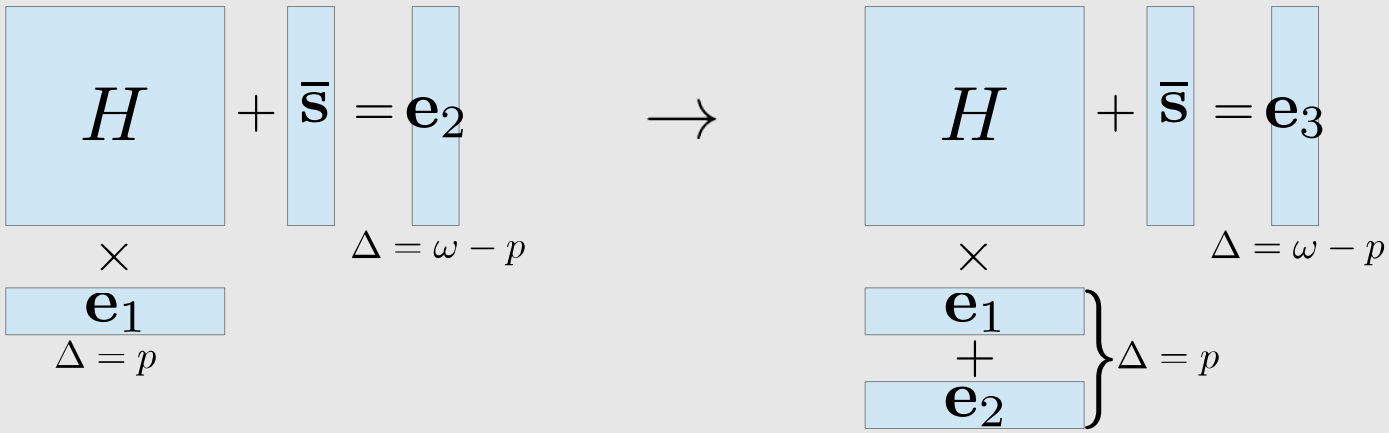
- Split the error vector again.



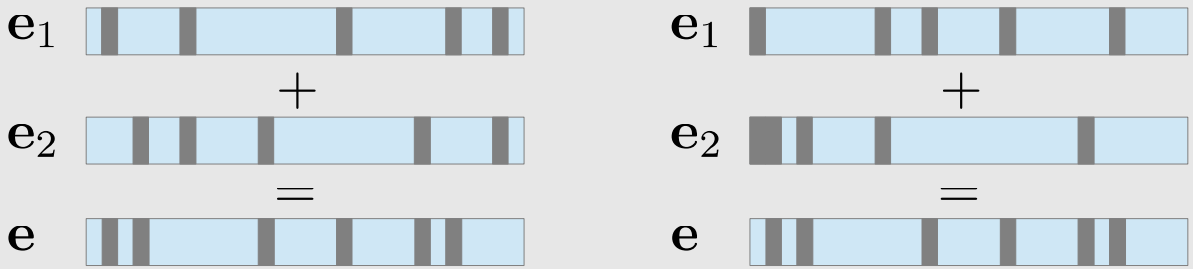
# Representations Techniques



- Split the error vector again.




- Many possible combinations create more solutions.



# Division into Blocks



- Solve equation blockwise.

# Division into Blocks



- Solve equation blockwise.


## Main Equations

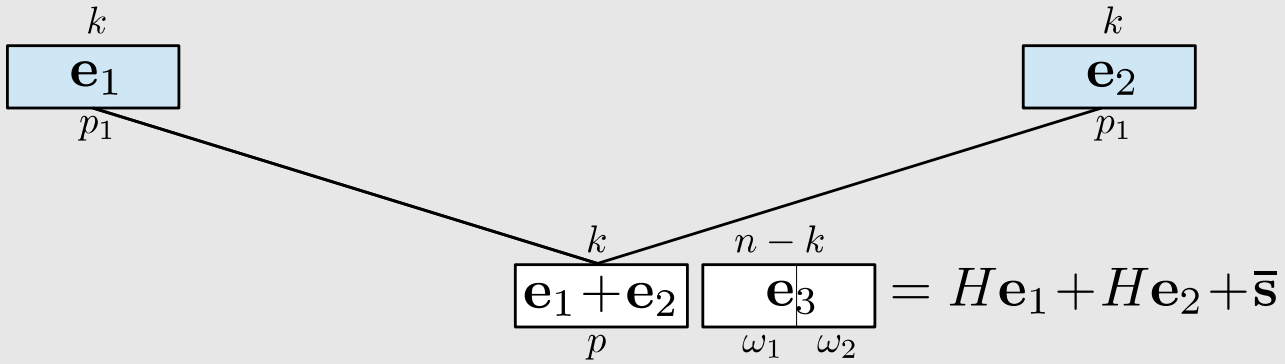
$$\text{"}\Delta(H\mathbf{e}_1 + H\mathbf{e}_2 + \bar{\mathbf{s}}) = \omega_1\text{" on the first block}$$
$$\text{"}\Delta(H\mathbf{e}_1 + H\mathbf{e}_2 + \bar{\mathbf{s}}) = \omega_2\text{" on the second block}$$

# Our Algorithm

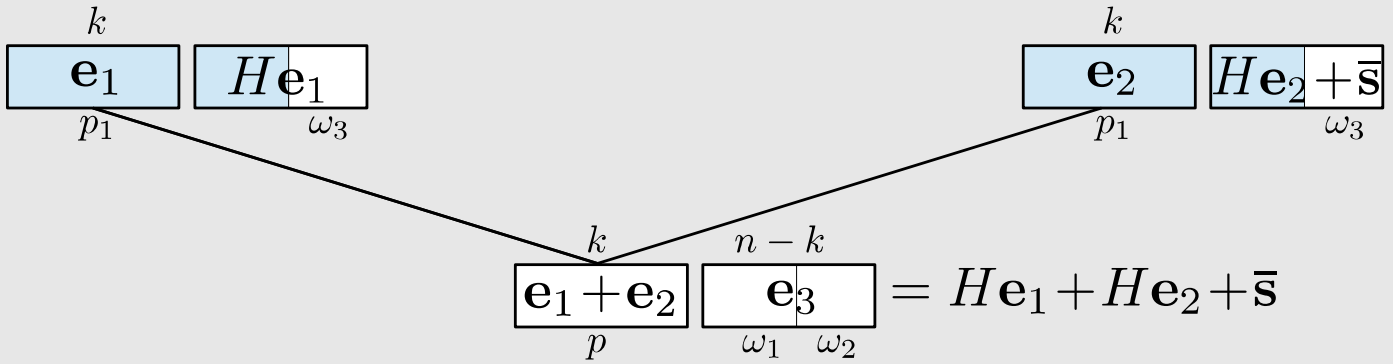
$$\overbrace{\boxed{\mathbf{e}_1+\mathbf{e}_2}}^{k}\overbrace{\boxed{\mathbf{e}_3}}^{n-k} = H\mathbf{e}_1+H\mathbf{e}_2+\bar{\mathbf{s}}$$

$p$ | $\omega_1$ | $\omega_2$

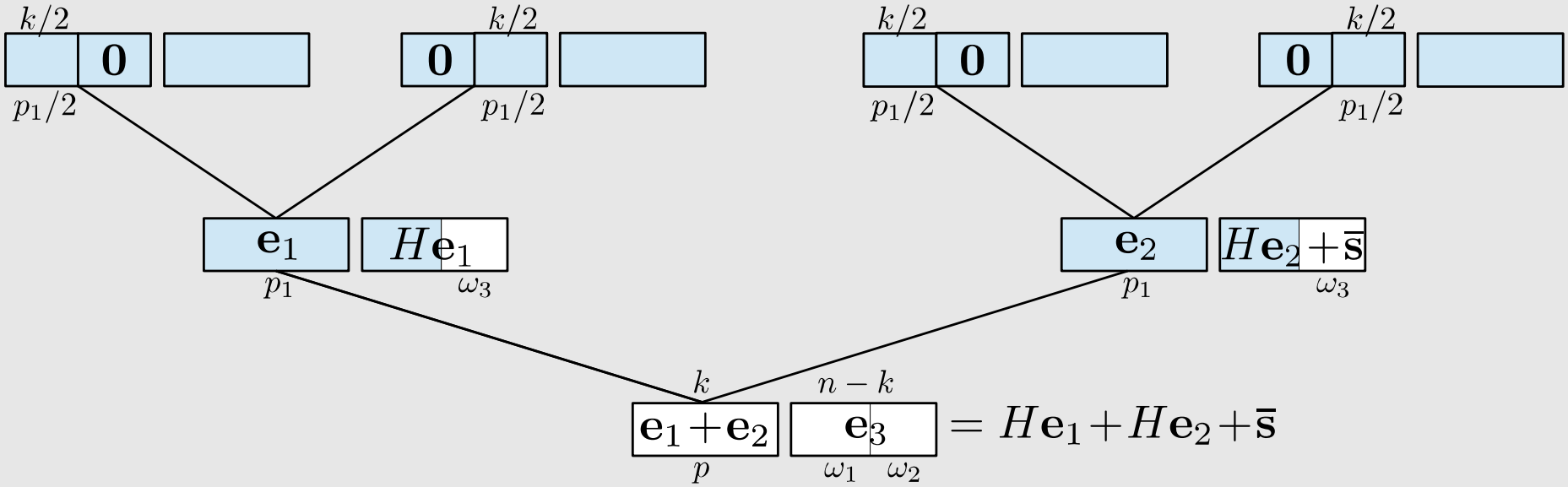
# Our Algorithm



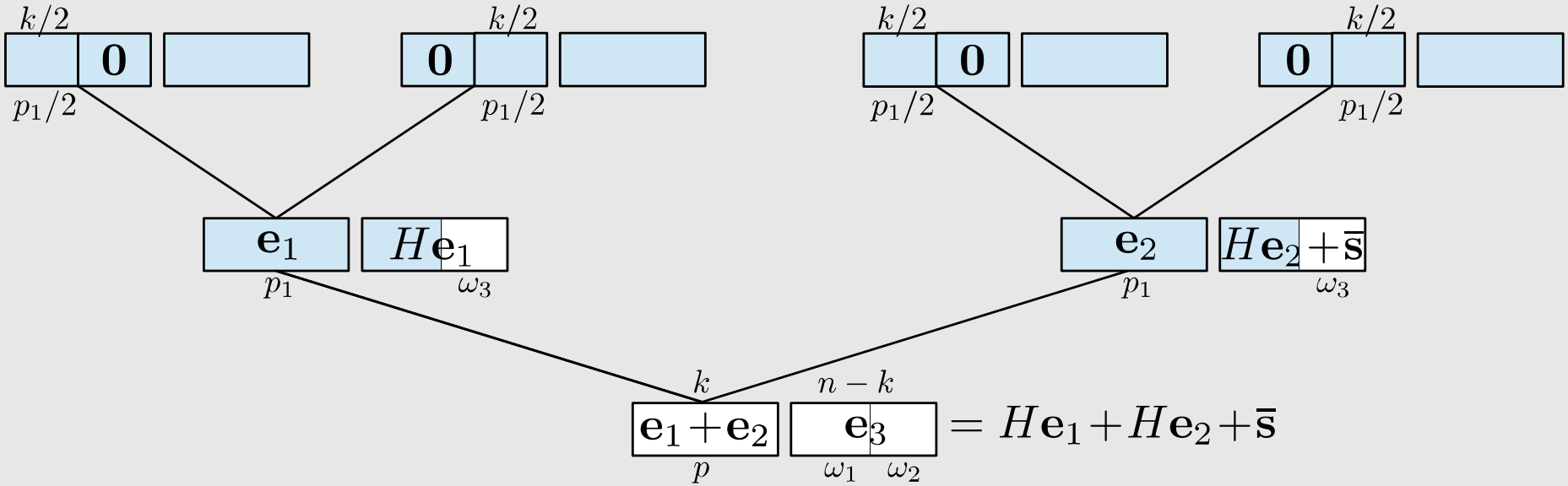
# Our Algorithm



# Our Algorithm



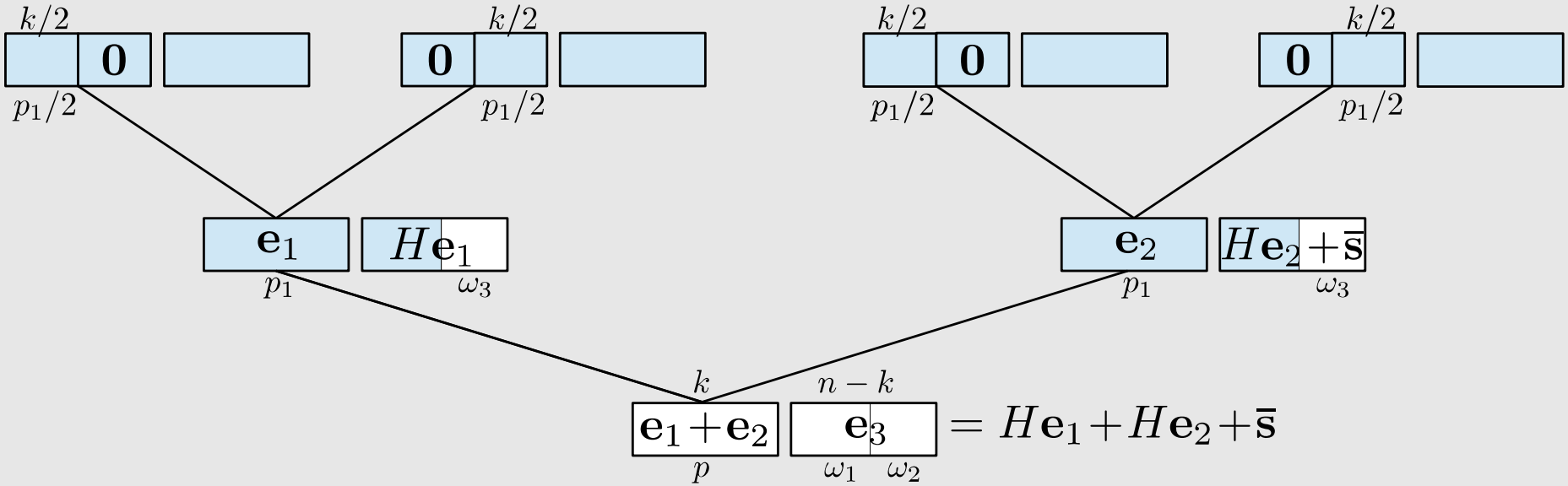
# Our Algorithm




## Algorithm (Idea)

1. Enumerate all vectors of length $k/2$ and weight $p_1/2$.

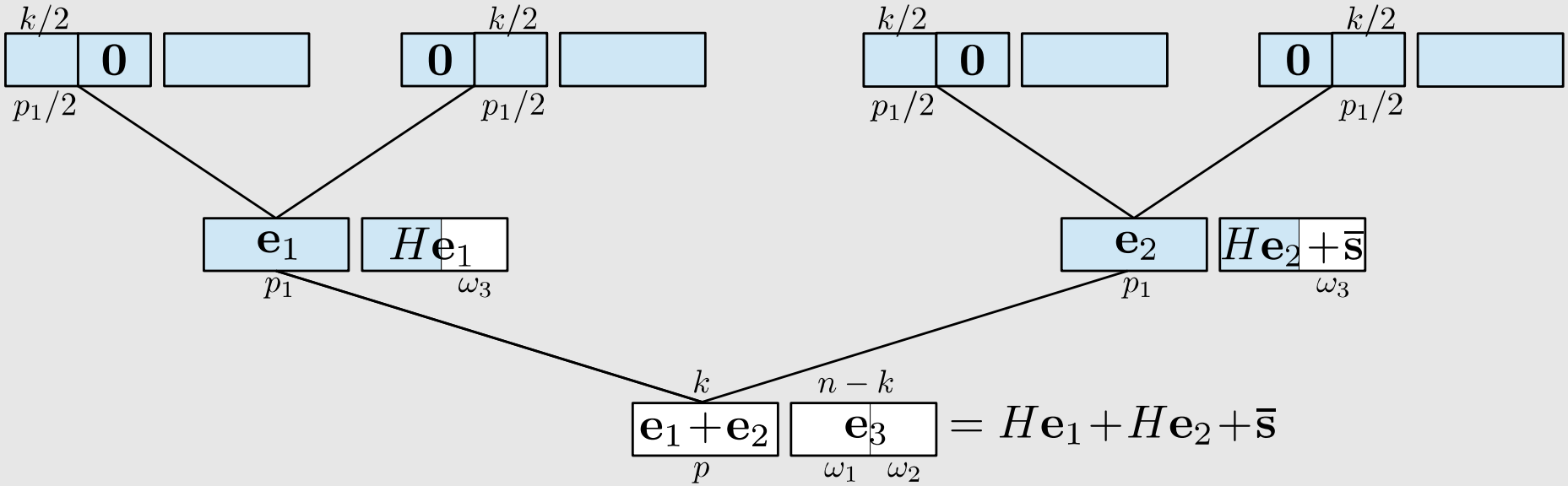
# Our Algorithm




## Algorithm (Idea)

1. Enumerate all vectors of length $k/2$ and weight $p_1/2$.
2. Nearest Neighbor search for weight $\omega_3$.

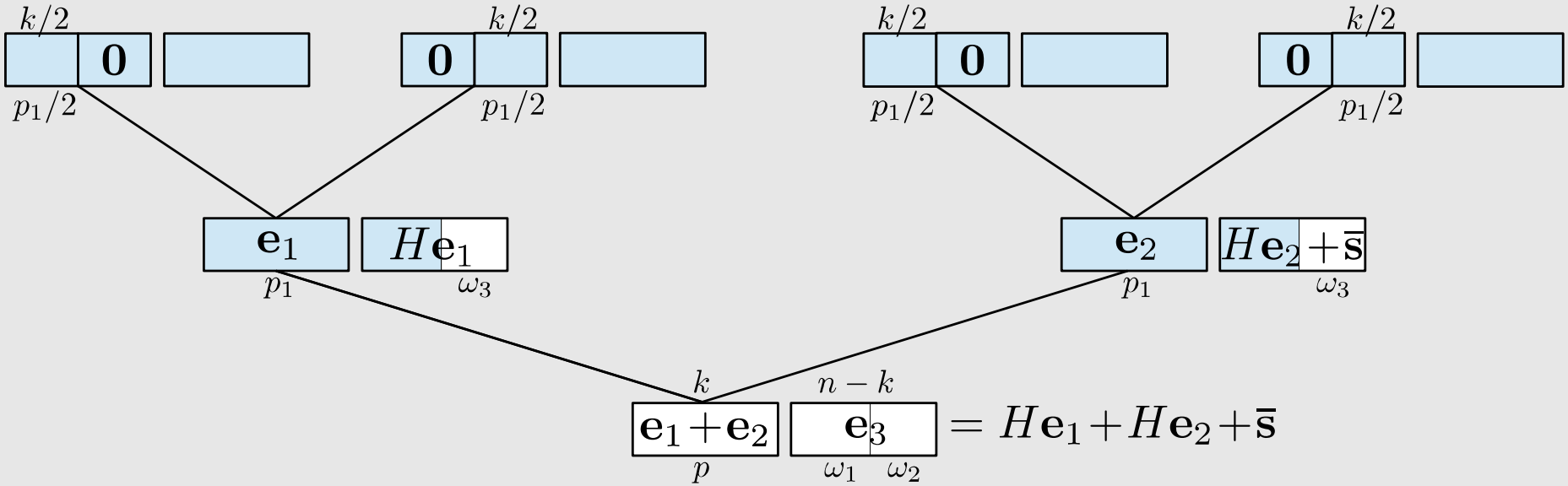
# Our Algorithm




## Algorithm (Idea)

1. Enumerate all vectors of length $k/2$ and weight $p_1/2$.
2. Nearest Neighbor search for weight $\omega_3$.
3. Nearest Neighbor search for weight $\omega_1$.

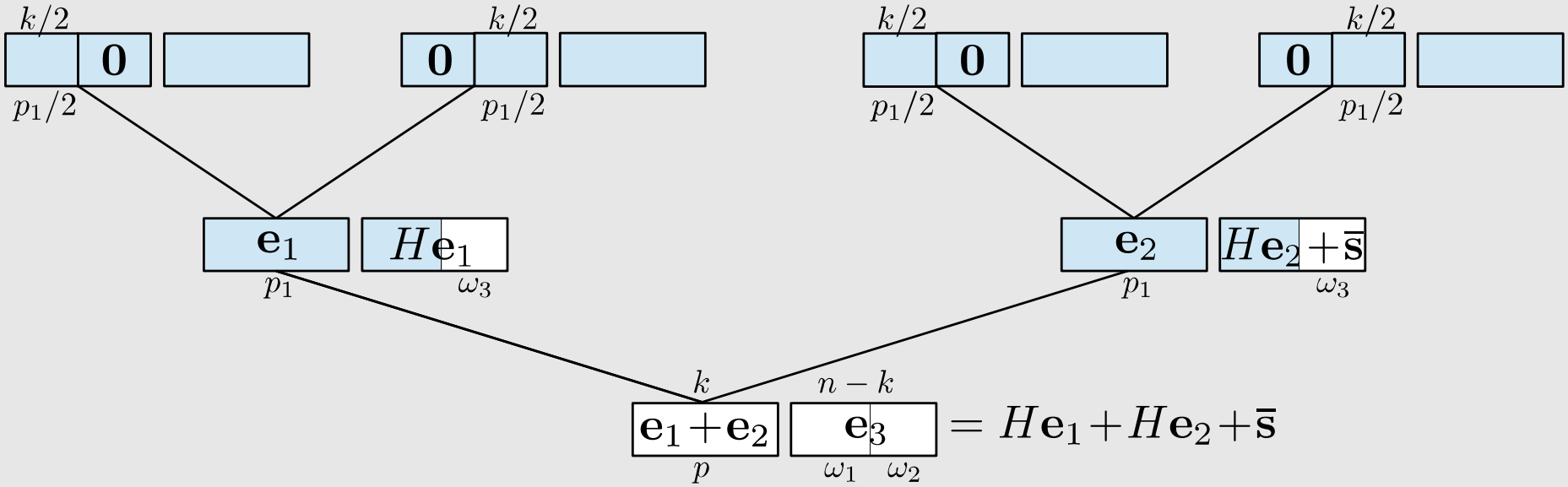
# Our Algorithm




## Algorithm (Idea)

1. Enumerate all vectors of length $k/2$ and weight $p_1/2$.
2. Nearest Neighbor search for weight $\omega_3$.
3. Nearest Neighbor search for weight $\omega_1$.
4. Filter for weight $p$, $\omega_2$.

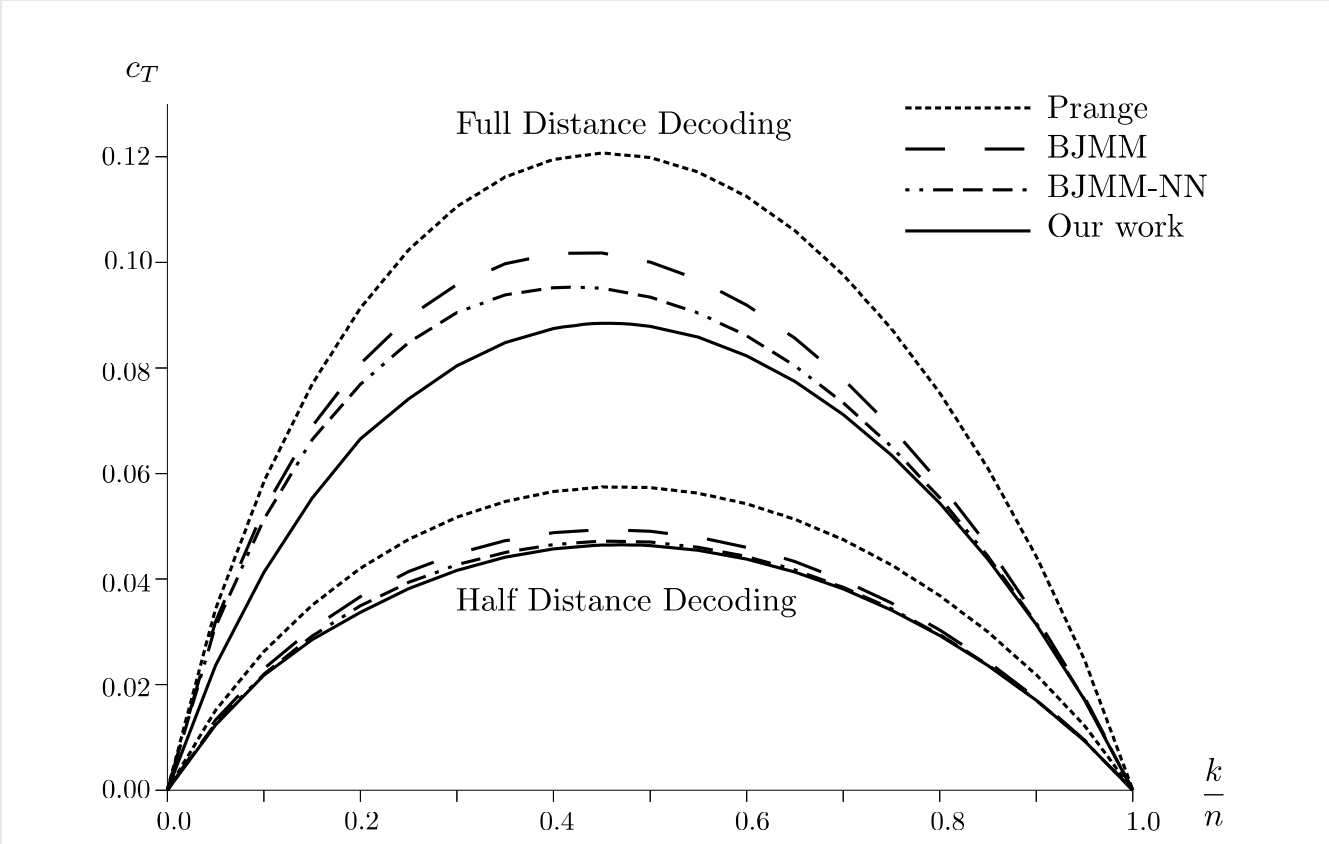
# Our Algorithm




- Can be generalized for an arbitrary number of levels.
- Uses May Ozerov Nearest Neighbor search whenever possible.
- Comparison to BJMM: NNS on every level, no exact matching.

# Results

- Comparison: Running time exponent $c_T$ for different code rates.



# Results

- Comparison: Running time exponent $c_T$ and memory exponent $c_M$ for different numbers of layers.

| | BJMM-NN | | Our algorithm | | |
|---|---|---|---|---|---|
| Layers | $c_T$ | $c_M$ | $c_T$ | $c_M$ | |
| 2 | 0.1003 | 0.0781 | 0.0982 | 0.0717 | |
| 3 | 0.0967 | 0.0879 | **0.0926** | 0.0647 | (FD) |
| 4 | **0.0953** | 0.0915 | **0.0885** | 0.0736 | |
| 2 | 0.0491 | 0.0309 | 0.0488 | 0.0290 | |
| 3 | 0.0473 | 0.0363 | 0.0478 | 0.0290 | (HD) |
| 4 | **0.0473** | 0.0351 | **0.0465** | 0.0294 | |

# Results

- Comparison: Running time exponent $c_T$ and memory exponent $c_M$ for different numbers of layers.

| | BJMM-NN | | Our algorithm | | |
|---|---|---|---|---|---|
| Layers | $c_T$ | $c_M$ | $c_T$ | $c_M$ | |
| 2 | 0.1003 | 0.0781 | 0.0982 | 0.0717 | |
| 3 | 0.0967 | 0.0879 | 0.0926 | 0.0647 | (FD) |
| 4 | 0.0953 | **0.0915** | 0.0885 | **0.0736** | |
| 2 | 0.0491 | 0.0309 | 0.0488 | 0.0290 | |
| 3 | 0.0473 | 0.0363 | 0.0478 | 0.0290 | (HD) |
| 4 | 0.0473 | **0.0351** | 0.0465 | **0.0294** | |

# Application: Hybrid Algorithm for LPN



## Definition ($\mathrm{LPN}_{k,\tau}$)

**Given:** Samples of the form

$$(\mathbf{a}_i, b_i) := (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i), \quad \text{for } i = 1, 2, \ldots$$

where $\mathbf{a}_i \in_R \mathbb{F}_2^k$ and $e_i \in \{0, 1\}$ with $\Pr[e_i = 1] = \tau \in [0, \frac{1}{2})$.

**Find:** $\mathbf{s} \in \mathbb{F}_2^k$.

- Alternative form: Write $n$ samples as

$$(A, \mathbf{b}) \in \mathbb{F}_2^{n \times k} \times \mathbb{F}_2^n \quad \text{satisfying } \mathbf{b} = A\mathbf{s} + \mathbf{e}.$$

- Connection to Decoding: $\mathbf{b}$ is noisy codeword.

# Application: Hybrid Algorithm for LPN



- Step 1: Use BKW algorithm to reduce dimension.
- Comes at cost of an increased error rate.

$$\mathrm{LPN}_{512,\frac{1}{4}} \rightarrow \mathrm{LPN}_{117,\frac{255}{512}}$$

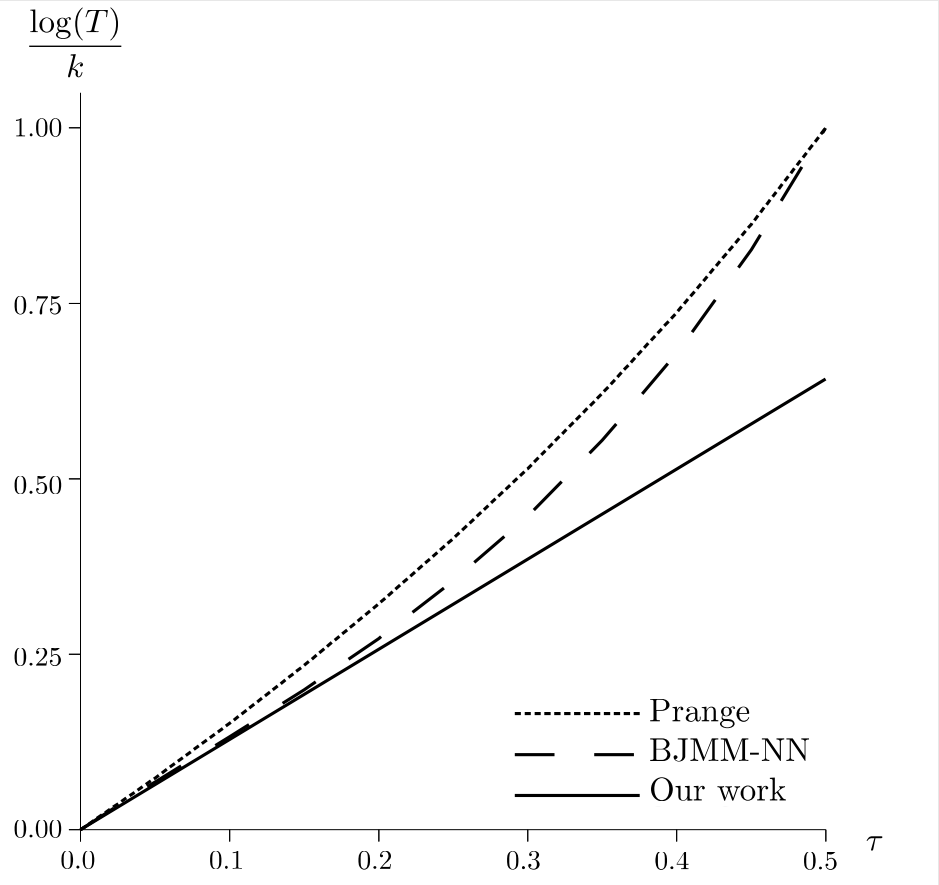
- Step 2: Solve instance via decoding.
- Comparison: Running time exponents for a typical instance

| $\mathrm{LPN}_{117,\frac{255}{512}}$ | $\log(T)$ | $\log(M)$ |
|---|---|---|
| Prange | 117 | - |
| BJMM-NN | 117 | 64 |
| Our algorithm | 75 | 47 |

# Application: Hybrid Algorithm for LPN



- Comparison: Running time exponent for different error rates.



# Summary



- Improved running times for decoding of random linear codes.
- Reason: Heavy use of Nearest Neighbor techniques.
- Superior algorithm in the hybrid framework for LPN.
- Open Problem: Reduce number of parameters.

Many thanks for your attention!

## Questions?