Sample Efficient Search to Decision for kLIN

Andrej Bogdanov	*University of Ottawa*
Alon Rosen	*Bocconi University*
Kel Zin Tan	*National University of Singapore*

kLIN (Sparse LPN)

kLIN (Sparse LPN)

Set sparsity $k = O(1)$, n variables , m equations

kLIN (Sparse LPN)

Set sparsity $k = O(1)$, n variables , m equations

Sample Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$, each row has exactly k non-zero entries

kLIN (Sparse LPN)

Set sparsity $k = O(1)$, n variables , m equations

Sample Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$, each row has exactly k non-zero entries

Sample Secret $s \leftarrow \mathbb{F}_2^n$

kLIN (Sparse LPN)

Set sparsity $k = O(1)$, n variables , m equations

Sample Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$, each row has exactly k non-zero entries

Sample Secret $s \leftarrow \mathbb{F}_2^n$

Sample Error $e \leftarrow \mathrm{Bern}(0.1)^m$

kLIN (Sparse LPN)

Set sparsity $k = O(1)$, n variables , m equations

Sample Matrix $A \leftarrow \mathbb{F}_2^{m\times n}$, each row has exactly k non-zero entries

Sample Secret $s \leftarrow \mathbb{F}_2^n$

Sample Error $e \leftarrow \mathrm{Bern}(0.1)^m$

Search: Recover s from $(A, As + e)$

kLIN (Sparse LPN)

Set sparsity $k = O(1)$, n variables , m equations

Sample Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$, each row has exactly k non-zero entries

Sample Secret $s \leftarrow \mathbb{F}_2^n$

Sample Error $e \leftarrow \text{Bern}(0.1)^m$

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u) where u is random binary vector

↑ Planted

↑ Null

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)
- Alternative assumption for PKE [ABW10]

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)
- Alternative assumption for PKE [ABW10]
- General field kLIN for advance crypto primitives [DIJL23, DJ24, CHKV24]

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)
- Alternative assumption for PKE [ABW10]
- General field kLIN for advance crypto primitives [DIJL23, DJ24, CHKV24]

General Predicate kLIN (Goldreich Function)

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)
- Alternative assumption for PKE [ABW10]
- General field kLIN for advance crypto primitives [DIJL23, DJ24, CHKV24]

General Predicate kLIN (Goldreich Function)

  - Search problem is a candidate one-way function (OWF) in NC0 [Gol00]

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)
- Alternative assumption for PKE [ABW10]
- General field kLIN for advance crypto primitives [DIJL23, DJ24, CHKV24]

General Predicate kLIN (Goldreich Function)

- Search problem is a candidate one-way function (OWF) in NC0 [Gol00]
- Decision problem is a pseudorandom generator (PRG)

Motivation

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

- kLIN is related to Constraint Satisfaction Problem (CSP)
- Alternative assumption for PKE [ABW10]
- General field kLIN for advance crypto primitives [DIJL23, DJ24, CHKV24]

General Predicate kLIN (Goldreich Function)

- Search problem is a candidate one-way function (OWF) in NC0 [Gol00]
- Decision problem is a pseudorandom generator (PRG)
- Search to Decision reduction implies a getting PRG from OWF in NC0

Hardness Comparison

Search: Recover s from $(A, As+e)$

Decision: Distinguish $(A, As+e)$ and (A, u)

Learning Parity With Noise (LPN)

Dense Matrix $A \leftarrow \mathbb{F}_2^{m\times n}$

Each entry in the matrix is a random bit

kLIN (Sparse LPN)

Sparse Matrix $A \leftarrow \mathbb{F}_2^{m\times n}$

Each row has exactly k non-zero entries

Hardness Comparison

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Learning Parity With Noise (LPN)

Dense Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each entry in the matrix is a random bit

Currently no polynomial time algorithm for search and decision for m = poly(n)

[BKW03, EKM17, AG11]

kLIN (Sparse LPN)

Sparse Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each row has exactly k non-zero entries

Hardness Comparison

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Learning Parity With Noise (LPN)

Dense Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each entry in the matrix is a random bit

Currently no polynomial time algorithm for search and decision for m = poly(n)

[BKW03, EKM17, AG11]

kLIN (Sparse LPN)

Sparse Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each row has exactly k non-zero entries

Polynomial time algorithm for both search and decision when

$$m = \Omega(n^{k/2})$$

Hardness Comparison

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Learning Parity With Noise (LPN)

Dense Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each entry in the matrix is a random bit

Currently no polynomial time algorithm for search and decision for m = poly(n)

[BKW03, EKM17, AG11]

kLIN (Sparse LPN)

Sparse Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each row has exactly k non-zero entries

Polynomial time algorithm for both search and decision when

$$m = \Omega(n^{k/2})$$

Since only $O(n^k)$ unique equations

Hardness Comparison

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Learning Parity With Noise (LPN)

Dense Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each entry in the matrix is a random bit

Currently no polynomial time algorithm for search and decision for m = poly(n)

[BKW03, EKM17, AG11]

kLIN (Sparse LPN)

Sparse Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each row has exactly k non-zero entries

Polynomial time algorithm for both search and decision when

$$m = \Omega(n^{k/2})$$

Since only $O(n^k)$ unique equations

Decision: Find collision

Hardness Comparison

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Learning Parity With Noise (LPN)

Dense Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each entry in the matrix is a random bit

Currently no polynomial time algorithm for search and decision for m = poly(n)

[BKW03, EKM17, AG11]

kLIN (Sparse LPN)

Sparse Matrix $A \leftarrow \mathbb{F}_2^{m \times n}$

Each row has exactly k non-zero entries

Polynomial time algorithm for both search and decision when

$$m = \Omega(n^{k/2})$$

Since only $O(n^k)$ unique equations

Decision: Find collision

Search: Non-trivial Idea [App16]

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Search To Decision

Search: Recover s from $(A, As+e)$

Decision: Distinguish $(A, As+e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search to Decision Reduction gives a partial answer

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search to Decision Reduction gives a partial answer

- LPN and LWE has **sample preserving** poly-time reduction [BIK09, MM11]

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search to Decision Reduction gives a partial answer

- LPN and LWE has **sample preserving** poly-time reduction [BIK09, MM11]
- Polynomial difference in the difficulty of Search and Decision

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search to Decision Reduction gives a partial answer

- LPN and LWE has **sample preserving** poly-time reduction [BIK09, MM11]
- Polynomial difference in the difficulty of Search and Decision

---

Since Search and Decision of kLIN both become viable when $m = \Omega(n^{k/2})$

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search to Decision Reduction gives a partial answer

- LPN and LWE has **sample preserving** poly-time reduction [BIK09, MM11]
- Polynomial difference in the difficulty of Search and Decision

---

Since Search and Decision of kLIN both become viable when $m = \Omega(n^{k/2})$

Plausible to believe **sample preserving** reduction exists for kLIN

Search To Decision

Search: Recover s from $(A, As + e)$

Decision: Distinguish $(A, As + e)$ and (A, u)

Search is harder than Decision

kLIN gets easier as number of equations m increases

Q: How does the difficulty of Search and Decision varies as m increases?

Search to Decision Reduction gives a partial answer

- LPN and LWE has **sample preserving** poly-time reduction [BIK09, MM11]
- Polynomial difference in the difficulty of Search and Decision

---

Since Search and Decision of kLIN both become viable when $m = \Omega(n^{k/2})$

Plausible to believe **sample preserving** reduction exists for kLIN

Gap: Improve the sample complexity

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high

$\xrightarrow{\text{Reduces}}$ kLin Decision
with m samples

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high
- [BSV19] $\tilde{O}(m^{2/k+1})$, success prob exponentially low

$\xrightarrow{\text{Reduces}}$

kLin Decision
with m samples

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high
- [BSV19] $\tilde{O}(m^{2/k+1})$, success prob exponentially low
- [BRT25] $O(nm)$, success prob high

$\xrightarrow{\text{Reduces}}$ kLin Decision
with m samples

$\xrightarrow{\text{Reduces}}$ (k-1)Lin Decision
with m samples

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high
- [BSV19] $\tilde{O}(m^{2/k+1})$, success prob exponentially low
- [BRT25] $O(nm)$, success prob high

$\xrightarrow{\text{Reduces}}$ kLin Decision with m samples

$\xrightarrow{\text{Reduces}}$ (k-1)Lin Decision with m samples

Assume that Search is hard when $m \ll n^{k/2}$

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high
- [BSV19] $\tilde{O}(m^{2/k+1})$, success prob exponentially low
- [BRT25] $O(nm)$, success prob high

$\xrightarrow{\text{Reduces}}$ kLin Decision
with m samples

$\xrightarrow{\text{Reduces}}$ (k-1)Lin Decision
with m samples

Assume that Search is hard when $m \ll n^{k/2}$

- [App12] implies kLin Decision is hard when $m \ll n^{k/6}$

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high
- [BSV19] $\tilde{O}(m^{2/k+1})$, success prob exponentially low
- [BRT25] $O(nm)$, success prob high

$\xrightarrow{\text{Reduces}}$ kLin Decision
with m samples

$\xrightarrow{\text{Reduces}}$ (k-1)Lin Decision
with m samples

Assume that Search is hard when $m \ll n^{k/2}$

- [App12] implies kLin Decision is hard when $m \ll n^{k/6}$
- [BRT25] implies (k-1)Lin Decision is hard when $m \ll n^{k/2-1}$

Previous Work & Results

Suppose Decision Algorithm D that has constant advantage

Search Problem with sample complexity

- [App12] $\tilde{O}(m^3)$, success prob high
- [BSV19] $\tilde{O}(m^{2/k+1})$, success prob exponentially low
- [BRT25] $O(nm)$, success prob high

$\xrightarrow{\text{Reduces}}$ kLin Decision with m samples

$\xrightarrow{\text{Reduces}}$ (k-1)Lin Decision with m samples

Assume that Search is hard when $m \ll n^{k/2}$

- [App12] implies kLin Decision is hard when $m \ll n^{k/6}$
- [BRT25] implies (k-1)Lin Decision is hard when $m \ll n^{k/2-1}$

Larger Stretch For PRG

Our Idea

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$

Our Idea

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$

Know this relationship for all $i \in [2, n]$ suffices for Search

Our Idea

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$

Know this relationship for all $i \in [2, n]$ suffices for Search

Use answer from Distinguisher D to find out

Our Idea

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$

Know this relationship for all $i \in [2, n]$ suffices for Search

Use answer from Distinguisher D to find out

Goal

- If $s_1 = s_i$, send Planted to D
- If $s_1 \neq s_i$, send Null to D

Our Idea

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$

Know this relationship for all $i \in [2, n]$ suffices for Search

Use answer from Distinguisher D to find out

Goal

- If $s_1 = s_i$, send Planted to D
- If $s_1 \neq s_i$, send Null to D

Since D can distinguish Planted and Null, it tells the relationship

Our Idea

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$

Know this relationship for all $i \in [2, n]$ suffices for Search

Use answer from Distinguisher D to find out

Goal

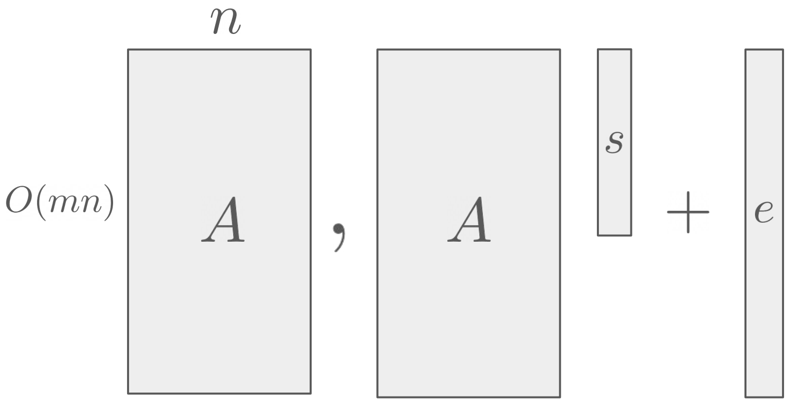
- If $s_1 = s_i$, send Planted to D
- If $s_1 \neq s_i$, send Null to D

Since D can distinguish Planted and Null, it tells the relationship

How to make this happen?

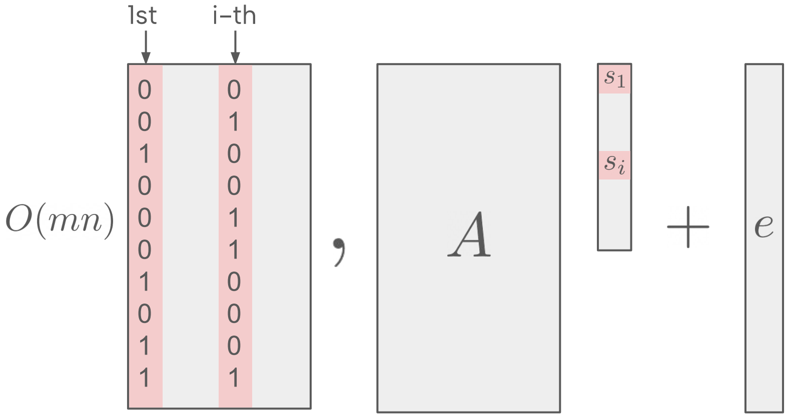
Transformation

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$ $A \in \mathbb{F}_2^{O(mn) \times n}$



Transformation

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$ $A \in \mathbb{F}_2^{O(mn) \times n}$

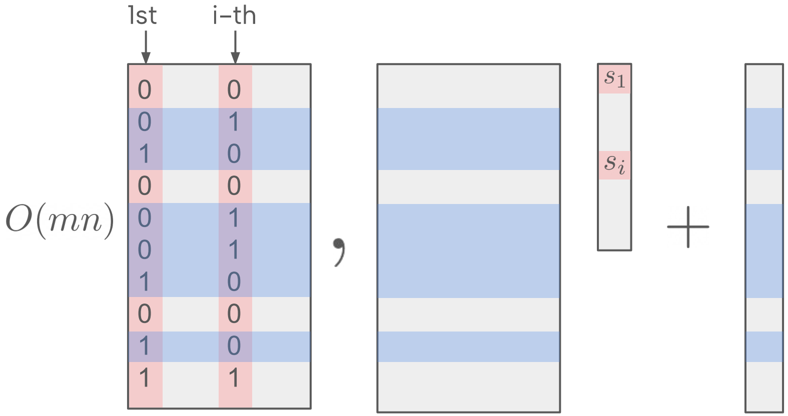


Look at 1st column and i-th column

Most of the entries in the columns will be zero due to sparsity

Transformation

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$ $A \in \mathbb{F}_2^{O(mn)\times n}$



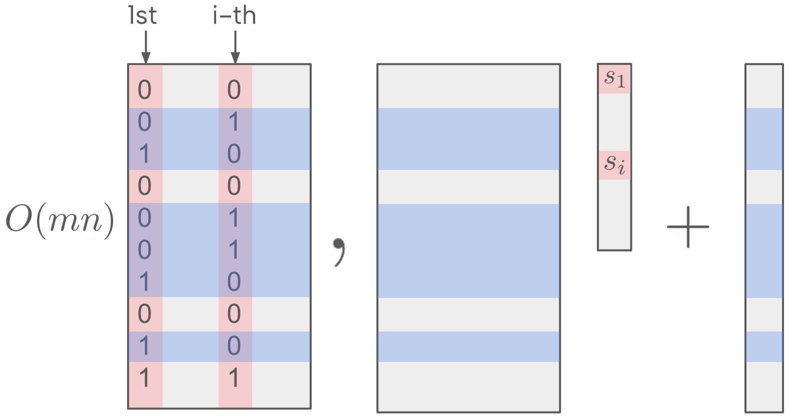
Collect the rows with distinct value at 1 and i

Look at 1st column and i-th column

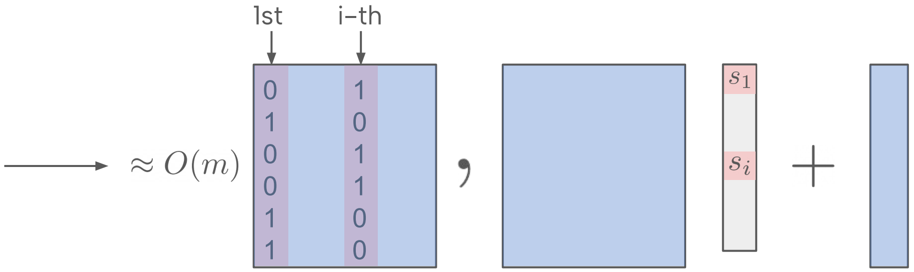
Most of the entries in the columns will be zero
due to sparsity

Transformation

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$ $A \in \mathbb{F}_2^{O(mn)\times n}$



Collect the rows with distinct value at 1 and i

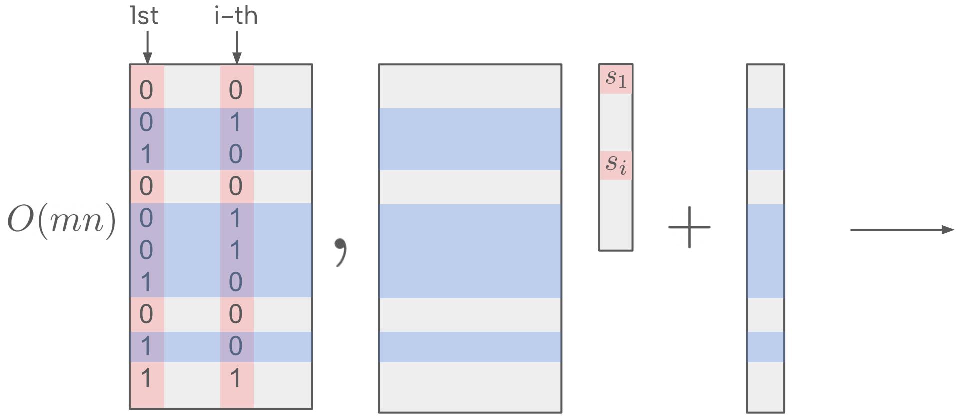


Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

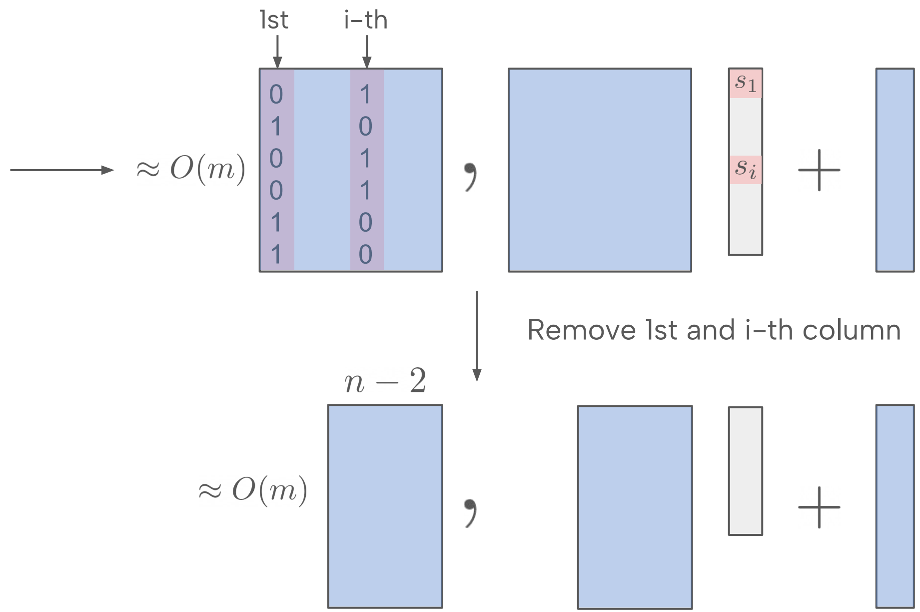
Transformation

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$ $A \in \mathbb{F}_2^{O(mn)\times n}$



Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

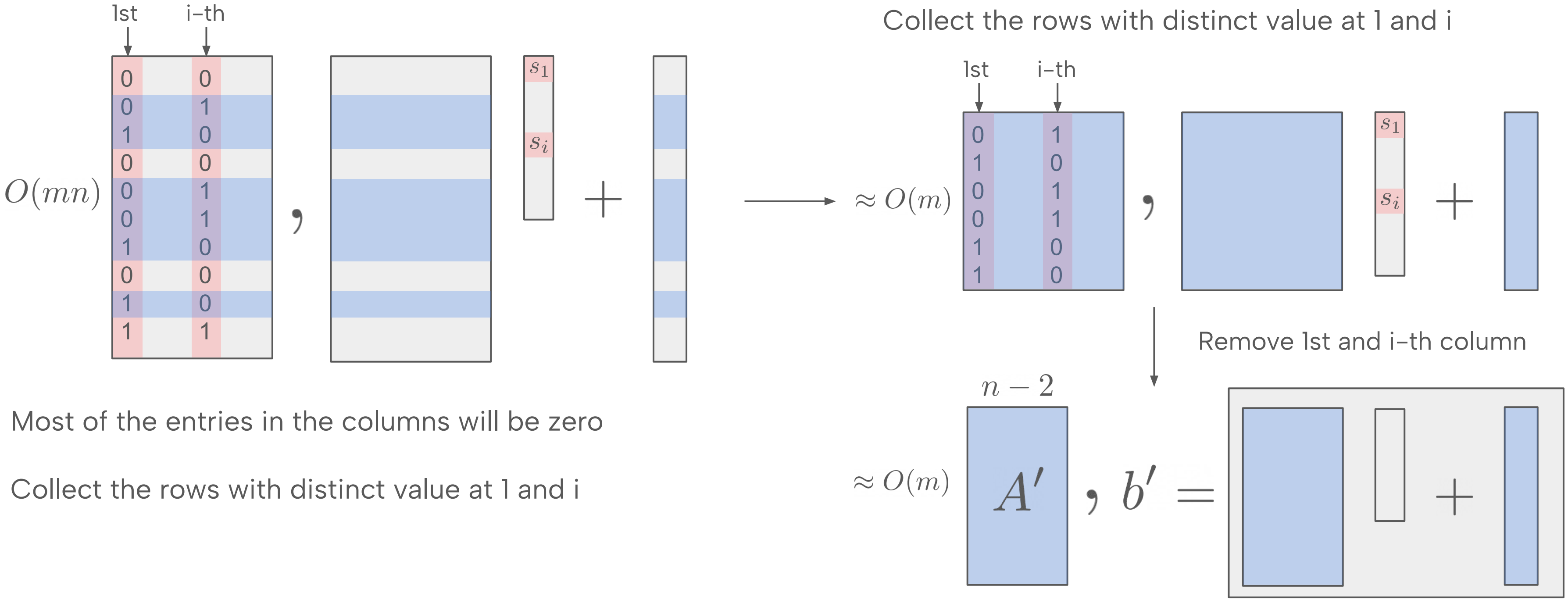


Collect the rows with distinct value at 1 and i

Remove 1st and i–th column

Transformation

We want to know whether $s_1 = s_i$ or $s_1 \neq s_i$ $A \in \mathbb{F}_2^{O(mn) \times n}$



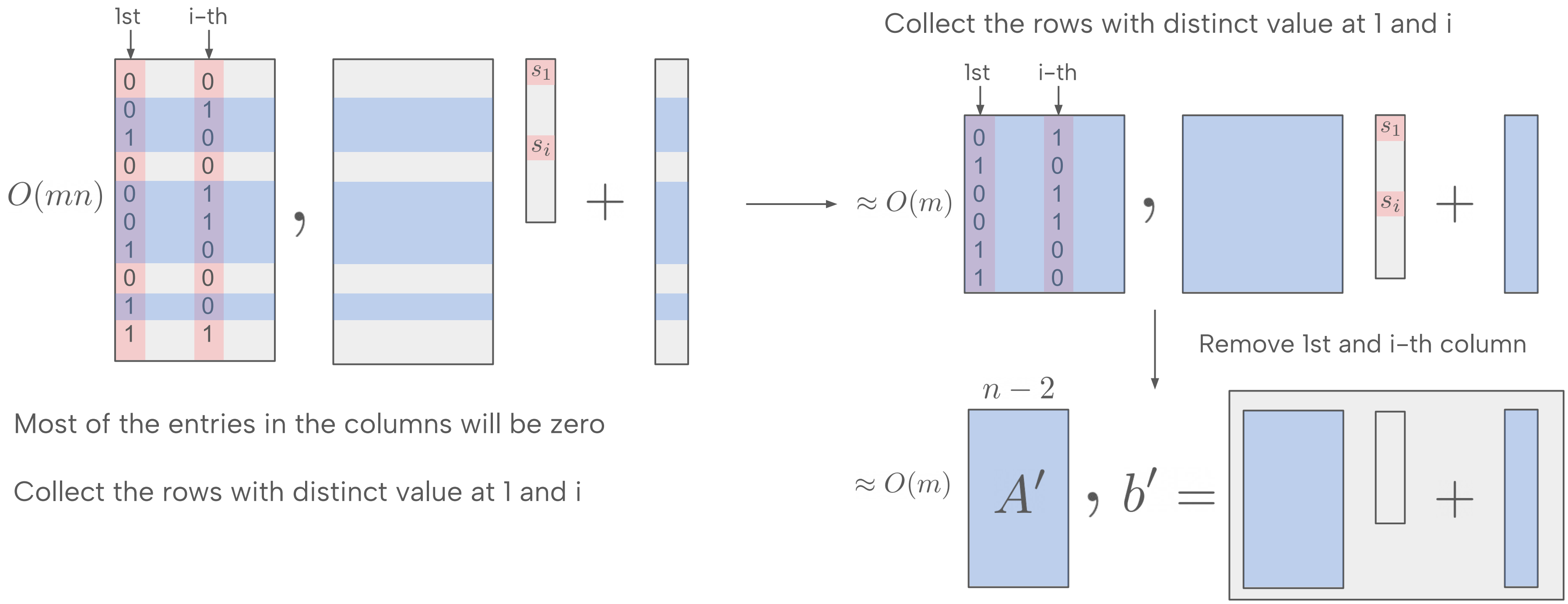
Collect the rows with distinct value at 1 and i

Remove 1st and i-th column

Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

Claim



Collect the rows with distinct value at 1 and i

Remove 1st and i-th column

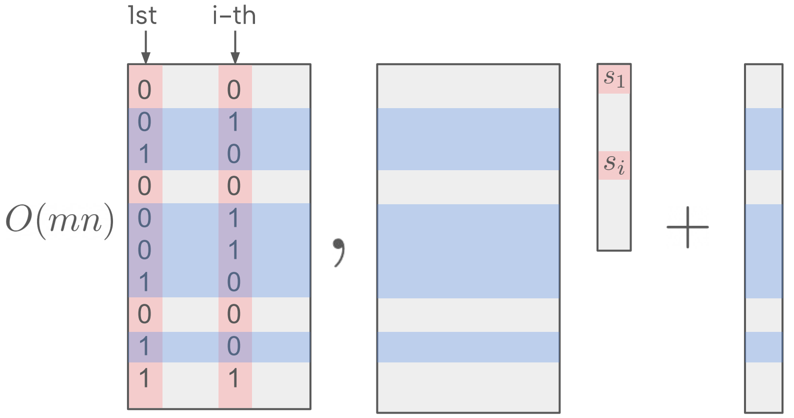
Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

Claim

- If $s_1 = s_i = 0$, then (A', b') distribute like Planted

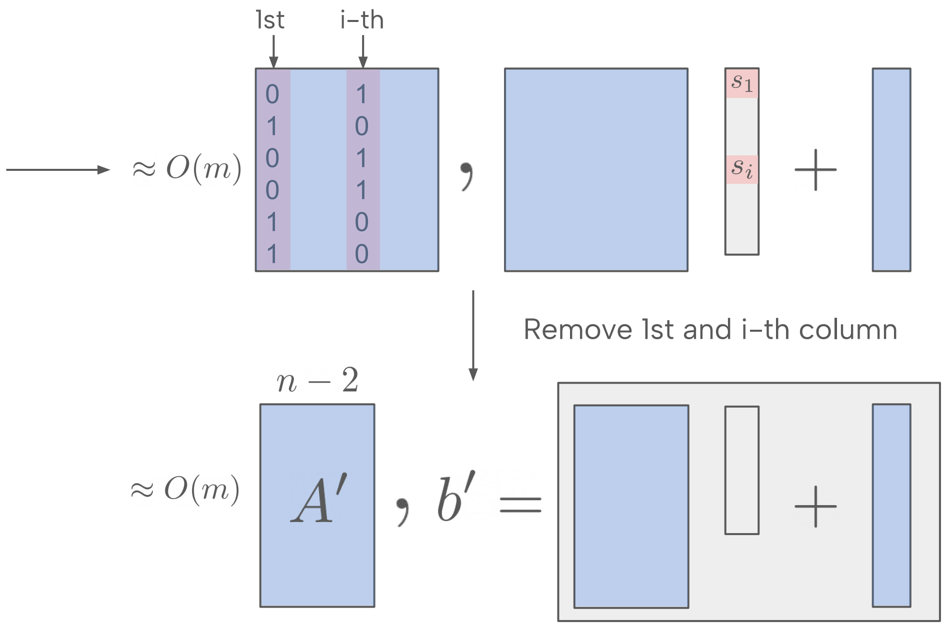
(k-1)Lin with (n-2) variables



Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

Collect the rows with distinct value at 1 and i

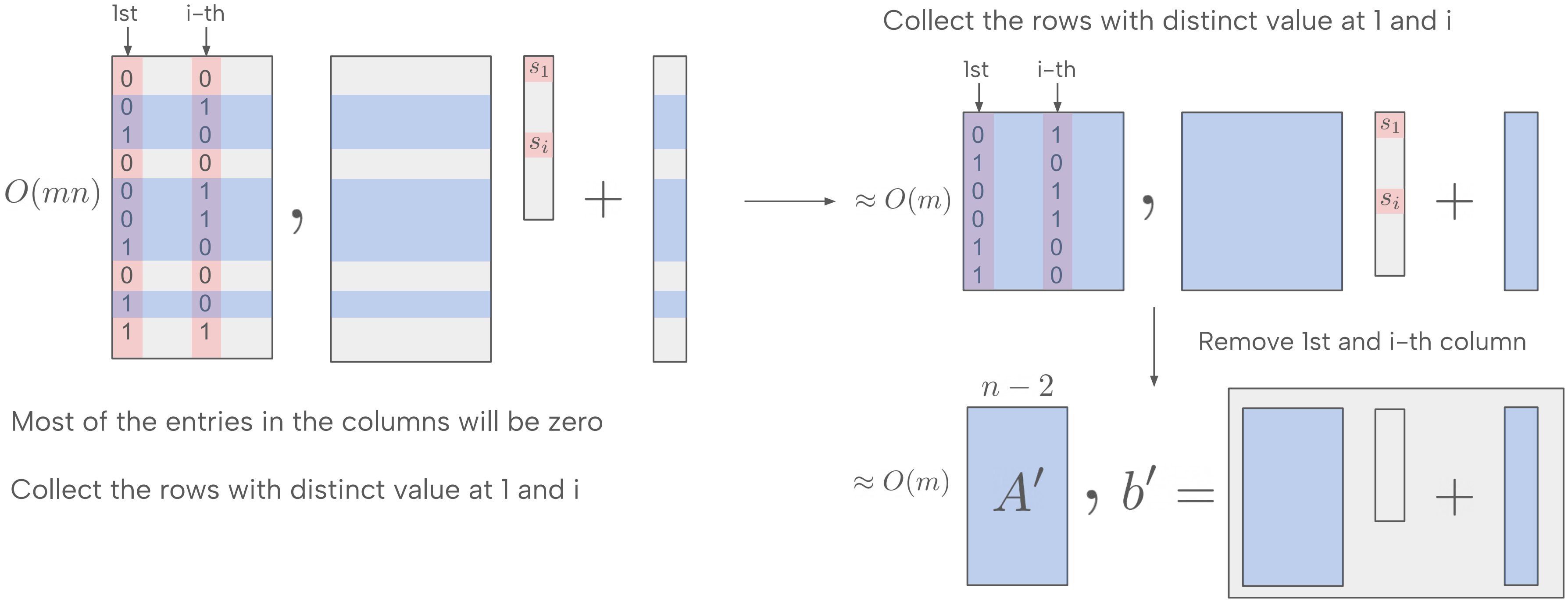


Remove 1st and i-th column

Claim

- If $s_1 = s_i = 0$, then (A', b') distribute like Planted
- If $s_1 = s_i = 1$, then $(A', \mathrm{flip}(b'))$ distribute like Planted

(k-1)Lin with (n-2) variables



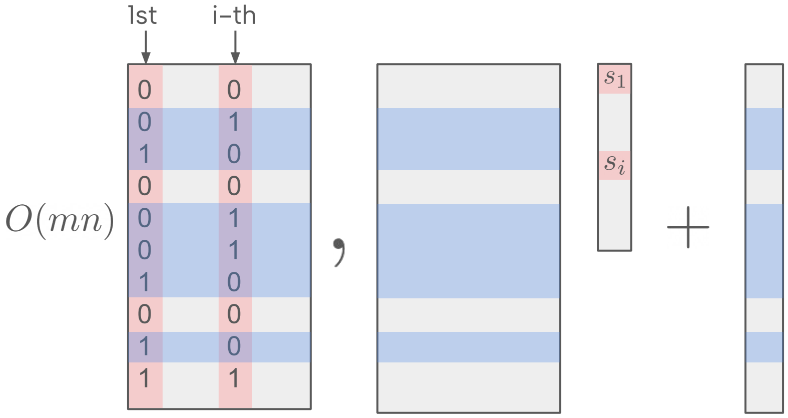
Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

Claim

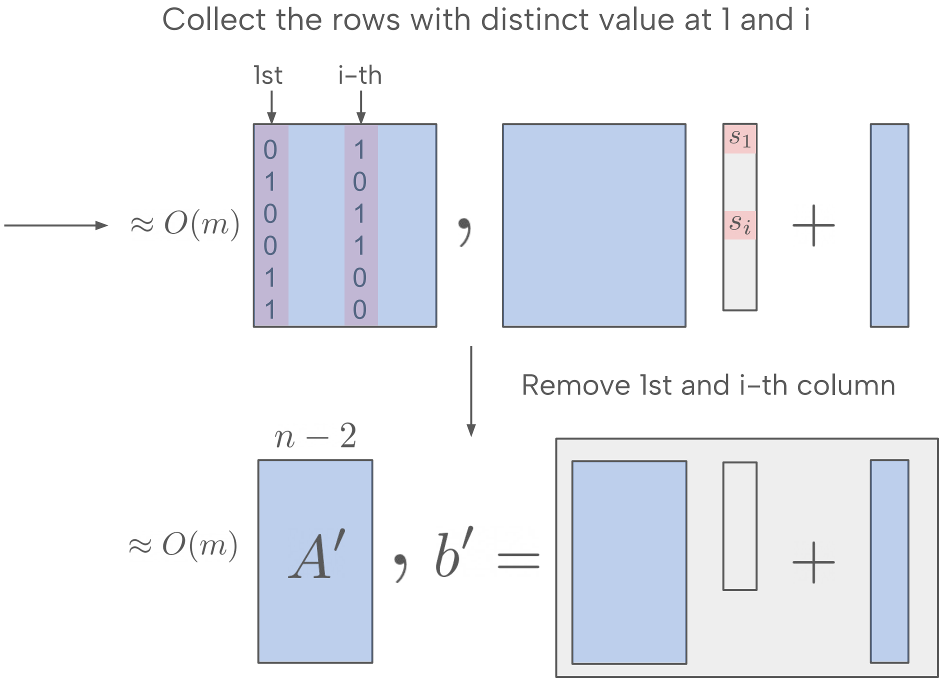
- If $s_1 = s_i = 0$, then (A', b') distribute like Planted
- If $s_1 = s_i = 1$, then $(A', \text{flip}(b'))$ distribute like Planted
- If $s_1 \neq s_i$, then (A', b'), $(A', \text{flip}(b'))$ distribute like Null

(k-1)Lin with (n-2) variables



Most of the entries in the columns will be zero

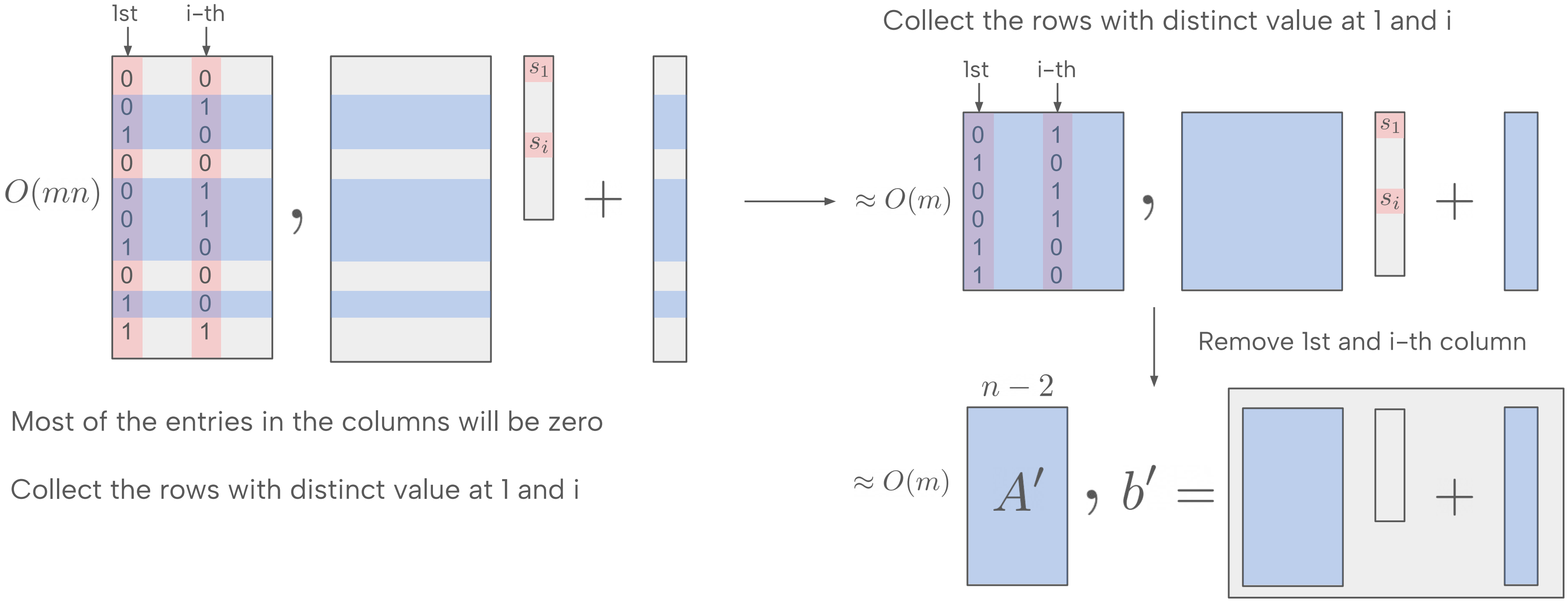
Collect the rows with distinct value at 1 and i



Claim

- If $s_1 = s_i = 0$, then (A', b') distribute like Planted
- If $s_1 = s_i = 1$, then $(A', \text{flip}(b'))$ distribute like Planted
- If $s_1 \neq s_i$, then (A', b'), $(A', \text{flip}(b'))$ distribute like Null

(k-1)Lin with (n-2) variables



Collect the rows with distinct value at 1 and i

Remove 1st and i-th column

Most of the entries in the columns will be zero

Collect the rows with distinct value at 1 and i

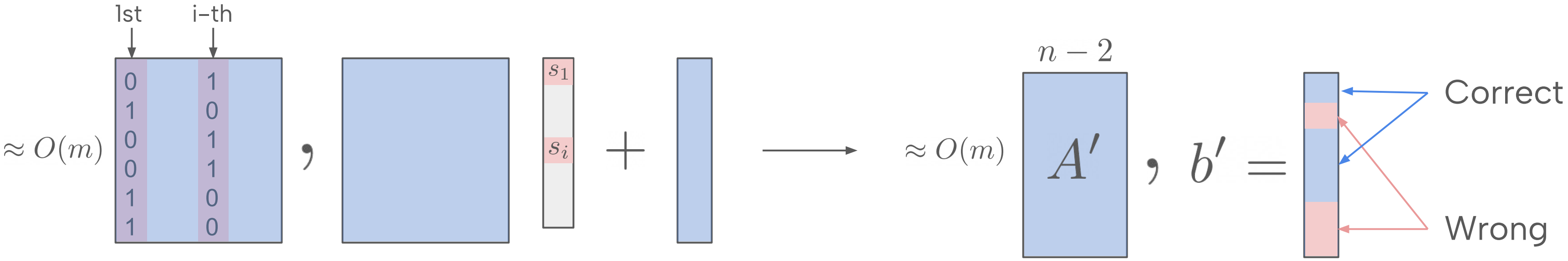
Claim

If $s_1 \neq s_i$, then (A', b'), $(A', \mathrm{flip}(b'))$ distribute like Null

Claim

If $s_1 \neq s_i$, then (A', b'), $(A', \text{flip}(b'))$ distribute like Null

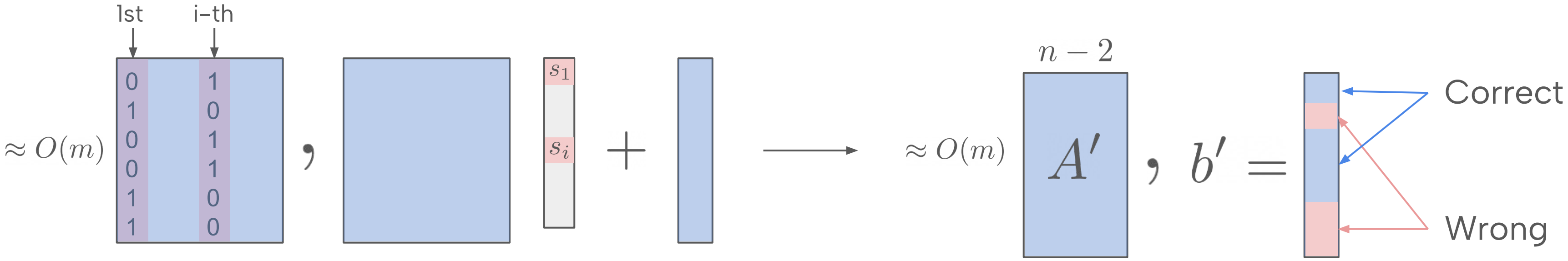
- If $s_1 = 1, s_i = 0$, (1 … 0 …) row is wrong, (0 … 1 …) row is correct



Claim

If $s_1 \neq s_i$, then (A', b'), $(A', \text{flip}(b'))$ distribute like Null

- If $s_1 = 1, s_i = 0$, (1 … 0 …) row is wrong, (0 … 1 …) row is correct
- If $s_1 = 0, s_i = 1$, (0 … 1 …) row is wrong, (1 … 0 …) row is correct

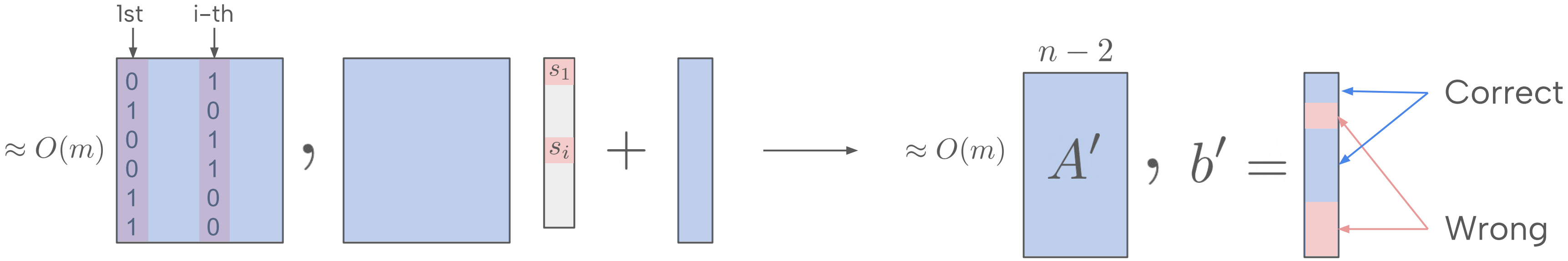


Claim

If $s_1 \neq s_i$, then (A', b'), $(A', \mathrm{flip}(b'))$ distribute like Null

- If $s_1 = 1, s_i = 0$, (1 … 0 …) row is wrong, (0 … 1 …) row is correct
- If $s_1 = 0, s_i = 1$, (0 … 1 …) row is wrong, (1 … 0 …) row is correct

On expectation, half of the rows (equations) are correct, other half is wrong



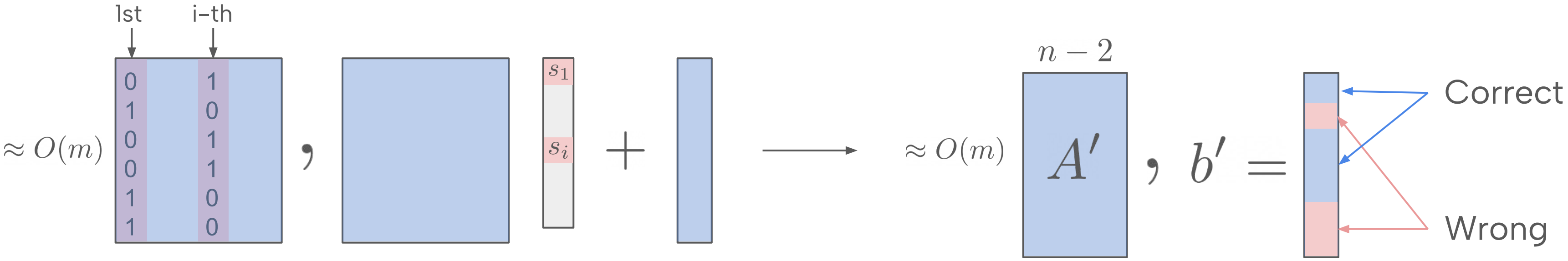
Claim

If $s_1 \neq s_i$, then (A', b'), $(A', \text{flip}(b'))$ distribute like Null

- If $s_1 = 1, s_i = 0$, (1 … 0 …) row is wrong, (0 … 1 …) row is correct
- If $s_1 = 0, s_i = 1$, (0 … 1 …) row is wrong, (1 … 0 …) row is correct

On expectation, half of the rows (equations) are correct, other half is wrong

D Does not know the origin of the rows, evaluated vector looks like a random vector



Algorithm

1. Guess value of s_1, predict s_i for all $i \in [2, n]$ by transform and send to D

Algorithm

1. Guess value of s_1, predict s_i for all $i \in [2, n]$ by transform and send to D
2. Independence can be enforced from not using same row, $O(mn)$ rows suffices

Algorithm

1. Guess value of s_1, predict s_i for all $i \in [2, n]$ by transform and send to D
2. Independence can be enforced from not using same row, $O(mn)$ rows suffices
3. Each prediction of s_i correct with some probability

Algorithm

1. Guess value of s_1 , predict s_i for all $i \in [2, n]$ by transform and send to D
2. Independence can be enforced from not using same row, $O(mn)$ rows suffices
3. Each prediction of s_i correct with some probability
4. Amplify the probability for s_i

Algorithm

1. Guess value of s_1, predict s_i for all $i \in [2, n]$ by transform and send to D
2. Independence can be enforced from not using same row, $O(mn)$ rows suffices
3. Each prediction of s_i correct with some probability
4. Amplify the probability for s_i

Amplification

Non-Error Aware

Error Aware

Algorithm

1. Guess value of s_1 , predict s_i for all $i \in [2, n]$ by transform and send to D
2. Independence can be enforced from not using same row, $O(mn)$ rows suffices
3. Each prediction of s_i correct with some probability
4. Amplify the probability for s_i

Amplification

Non-Error Aware

1. Get more equations
2. Repeat procedure
3. Take majority answer

Error Aware

Algorithm

1. Guess value of s_1 , predict s_i for all $i \in [2, n]$ by transform and send to D
2. Independence can be enforced from not using same row, $O(mn)$ rows suffices
3. Each prediction of s_i correct with some probability
4. Amplify the probability for s_i

Amplification

Non-Error Aware

1. Get more equations
2. Repeat procedure
3. Take majority answer

Error Aware

1. Get more equations
2. Substitute in predicted secret
3. Get more accurate secret
4. Repeat

Generalization

Larger Finite Field

Our Technique works for Sparse $\mathbb{F}_q$- LPN

Sample size independent from size of Field, but time complexity affected

Generalization

Larger Finite Field

Our Technique works for Sparse $\mathbb{F}_q$- LPN

Sample size independent from size of Field, but time complexity affected

Generalization

Larger Finite Field

Our Technique works for Sparse $\mathbb{F}_q$ - LPN

Sample size independent from size of Field, but time complexity affected

Goldreich Function

Model kLin as $P(s_{i_1}, s_{i_2}, \ldots, s_{i_k}) = s_{i_1} + s_{i_2} + \cdots + s_{i_k} + e$

Generalization

Larger Finite Field

Our Technique works for Sparse $\mathbb{F}_q$- LPN

Sample size independent from size of Field, but time complexity affected

Goldreich Function

Model kLin as $P(s_{i_1}, s_{i_2}, \ldots, s_{i_k}) = s_{i_1} + s_{i_2} + \cdots + s_{i_k} + e$

Our technique works for as long as $P(s_{i_1}, s_{i_2}, \ldots, s_{i_k}) = s_{i_1} + Q(s_{i_2}, \ldots, s_{i_{k-1}})$

For arbitrary $\mathbb{F}_2$ valued function Q

Assume that Decision Algorithm is secret independent

Concluding Remarks

Currently, efficient algorithm exists only with sample $\Omega(n^{k/2})$

Concluding Remarks

Currently, efficient algorithm exists only with sample $\Omega(n^{k/2})$

- Best (k-1)Lin decision algorithm requires $\Omega(n^{(k-1)/2})$
- Best kLin search algorithm requires $\Omega(n^{k/2})$

Concluding Remarks

Currently, efficient algorithm exists only with sample $\Omega(n^{k/2})$

- Best (k–1)Lin decision algorithm requires $\Omega(n^{(k-1)/2})$
- Best kLin search algorithm requires $\Omega(n^{k/2})$

Our reduction shows

Best efficient
(k–1)Lin Decision $\Omega(n^{(k-1)/2})$

implies

Exist efficient kLin
Search $\Omega(\underset{m}{\underline{n^{(k-1)/2}}} \times n) = \Omega(n^{(k+1)/2})$

Concluding Remarks

Currently, efficient algorithm exists only with sample $\Omega(n^{k/2})$

- Best (k–1)Lin decision algorithm requires $\Omega(n^{(k-1)/2})$
- Best kLin search algorithm requires $\Omega(n^{k/2})$

Our reduction shows

Best efficient
(k–1)Lin Decision $\Omega(n^{(k-1)/2})$ $\xrightarrow{\text{implies}}$ Exist efficient kLin
Search $\Omega(\underset{m}{\underline{n^{(k-1)/2}}} \times n) = \Omega(n^{(k+1)/2})$

$\updownarrow$

However,
Best kLin
Search $\Omega(n^{k/2})$

Concluding Remarks

Currently, efficient algorithm exists only with sample $\Omega(n^{k/2})$

- Best (k–1)Lin decision algorithm requires $\Omega(n^{(k-1)/2})$
- Best kLin search algorithm requires $\Omega(n^{k/2})$

Our reduction shows

Best efficient
(k–1)Lin Decision $\Omega(n^{(k-1)/2})$ $\xrightarrow{\text{implies}}$ Exist efficient kLin
Search $\Omega(\underset{m}{\underline{n^{(k-1)/2}}} \times n) = \Omega(n^{(k+1)/2})$

$\updownarrow$ Gap $\sqrt{n}$ from optimal

However,
Best kLin
Search $\Omega(n^{k/2})$

Concluding Remarks

Currently, efficient algorithm exists only with sample $\Omega(n^{k/2})$

- Best (k–1)Lin decision algorithm requires $\Omega(n^{(k-1)/2})$
- Best kLin search algorithm requires $\Omega(n^{k/2})$

Our reduction shows

Best efficient (k–1)Lin Decision $\Omega(n^{(k-1)/2})$ $\xrightarrow{\text{implies}}$ Exist efficient kLin Search $\Omega(\underbrace{n^{(k-1)/2}}_{m} \times n) = \Omega(n^{(k+1)/2})$

$\updownarrow$ Gap $\sqrt{n}$ from optimal

However, Best kLin Search $\Omega(n^{k/2})$

Open Question: Improve the reduction by a square root factor

Thank You