Indistinguishability Obfuscation from Bilinear Maps and LPN Variants

TCC 2024

December 6, 2024

Seyoon Ragavan
MIT

Neekon Vafa
MIT

Vinod Vaikuntanathan
MIT

Indistinguishability Obfuscation

PPT algorithm $\mathcal{O}$, where inputs and outputs are circuits.

Indistinguishability Obfuscation

PPT algorithm $\mathcal{O}$, where inputs and outputs are circuits.

- **Correctness**: $\forall$ circuits C, inputs x, $\mathcal{O}(C)(x) = C(x)$.

Indistinguishability Obfuscation

PPT algorithm $\mathcal{O}$, where inputs and outputs are circuits.

- **Correctness**: $\forall$ circuits C, inputs x, $\mathcal{O}(C)(x) = C(x)$.
- **Indistinguishability Security**: For all same-size, functionally equivalent circuits C_0, C_1,

$$\mathcal{O}\left(C_0\right) \approx_c \mathcal{O}\left(C_1\right).$$

IO is “crypto-complete”

IO (+ other mild assumptions) implies:

IO is "crypto-complete"

IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),

IO is “crypto-complete”

IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),
- … much more.

IO is "crypto-complete"

IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),
- … much more.

Can we build it?

IO is “crypto-complete”

IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),
- … much more.

Can we build it?

2013 - 2020:

IO is “crypto-complete”

IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),
- … much more.

Can we build it?

2013 - 2020: …maybe?

IO is "crypto-complete"

IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),
- … much more.

Can we build it?

2013 - 2020: …maybe?

2021: Jain, Lin, and Sahai:

IO is “crypto-complete”

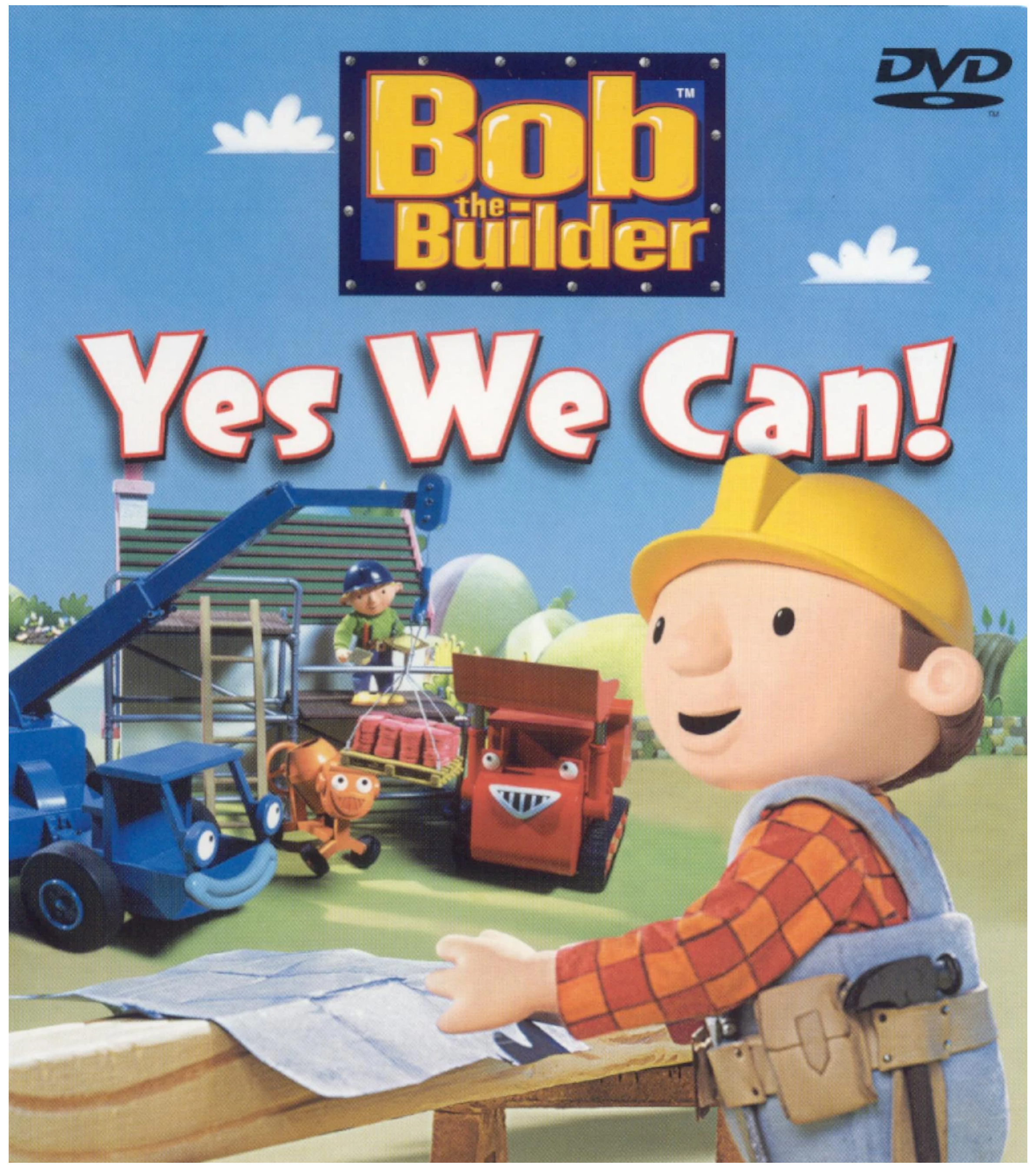
IO (+ other mild assumptions) implies:

- Fully homomorphic encryption,
- ZK-SNARGs for NP,
- Functional encryption (FE),
- … much more.

Can we build it?

2013 - 2020: …maybe?

2021: Jain, Lin, and Sahai:



IO From Well-Founded Assumptions

Theorem [JLS ’21]: Construction of IO from (sub-exponential)

IO From Well-Founded Assumptions

Theorem [JLS '21]: Construction of IO from (sub-exponential)

- $\boxed{\text{Learning With Errors (LWE)}}$,

IO From Well-Founded Assumptions

Theorem [JLS '21]: Construction of IO from (sub-exponential)

- $\boxed{\text{Learning With Errors (LWE)}}$,
- $\boxed{\text{Bilinear Maps (SXDH)}}$,



IO From Well-Founded Assumptions

Theorem [JLS '21]: Construction of IO from (sub-exponential)

- Learning With Errors (LWE),
- Bilinear Maps (SXDH),
- (Large-field) Learning Parity with Noise (LPN), &



IO From Well-Founded Assumptions

Theorem [JLS '21]: Construction of IO from (sub-exponential)

- Learning With Errors (LWE) ,
- Bilinear Maps (SXDH),
- (Large-field) Learning Parity with Noise (LPN) , &
- PRGs in NC^0 with polynomial stretch .



IO From Well-Founded Assumptions

Theorem [JLS '**22**]: Construction of IO from (sub-exponential)

- ~~*Learning With Errors (LWE)*~~ ,
- Bilinear Maps (DLIN) ,
- (Large-field) Learning Parity with Noise (LPN) , &
- PRGs in NC^0 with polynomial stretch .



Our Result

Theorem [RVV '24]: Construction of IO from (sub-exponential)

- ~~*Learning With Errors (LWE)*~~ ,
- Bilinear Maps (DLIN) ,
- (Large-field) Learning Parity with Noise (LPN) , &
- ~~*PRGs in* NC^0 *with polynomial stretch*~~ **"Local Functions with Noise" (LFN)**

Our Result

Theorem [RVV '24]: Construction of IO from (sub-exponential)

- *~~Learning With Errors (LWE)~~* ,
- Bilinear Maps (DLIN) ,
- (Large-field) Learning Parity with Noise (LPN) , &
- *~~PRGs in NC^0 with polynomial stretch~~* **"Local Functions with Noise" (LFN)**

Weaker!

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

$$\left(f, f(\mathbf{s}) \oplus \mathbf{e}\right) \approx_c \left(f, \$\right).$$

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

$$\left(f, f(\mathbf{s}) \oplus \mathbf{e}\right) \approx_c \left(f, \$\right).$$

$$\mathbf{s} \leftarrow \mathbb{Z}_2^n$$
$$\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$$

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

Each output bit depends on $O(1)$ input bits.

$$\left(f, f(\mathbf{s}) \oplus \mathbf{e}\right) \approx_c \left(f, \$\right).$$

$$\mathbf{s} \leftarrow \mathbb{Z}_2^n$$
$$\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$$

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

Each output bit depends on $O(1)$ input bits.

$$\left(f, f(\mathbf{s}) \oplus \mathbf{e}\right) \approx_c \left(f, \$\right).$$

$$\mathbf{s} \leftarrow \mathbb{Z}_2^n$$
$$\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$$

Two ways to instantiate LFN:

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

Each output bit depends on $O(1)$ input bits.

$$\left(f, f(\mathbf{s}) \oplus \mathbf{e}\right) \approx_c \left(f, \$\right).$$

$$\mathbf{s} \leftarrow \mathbb{Z}_2^n$$
$$\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$$

Two ways to instantiate LFN:

1. PRGs in NC^0 (e.g., Goldreich's PRGs): no noise! ($\delta \to \infty$).

Local Functions with Noise (LFN)

There is a distribution over NC^0 functions $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$, with $m = n^{1+\varepsilon}$,

Each output bit depends on $O(1)$ input bits.

$$\left(f, f(\mathbf{s}) \oplus \mathbf{e}\right) \approx_c \left(f, \$\right).$$

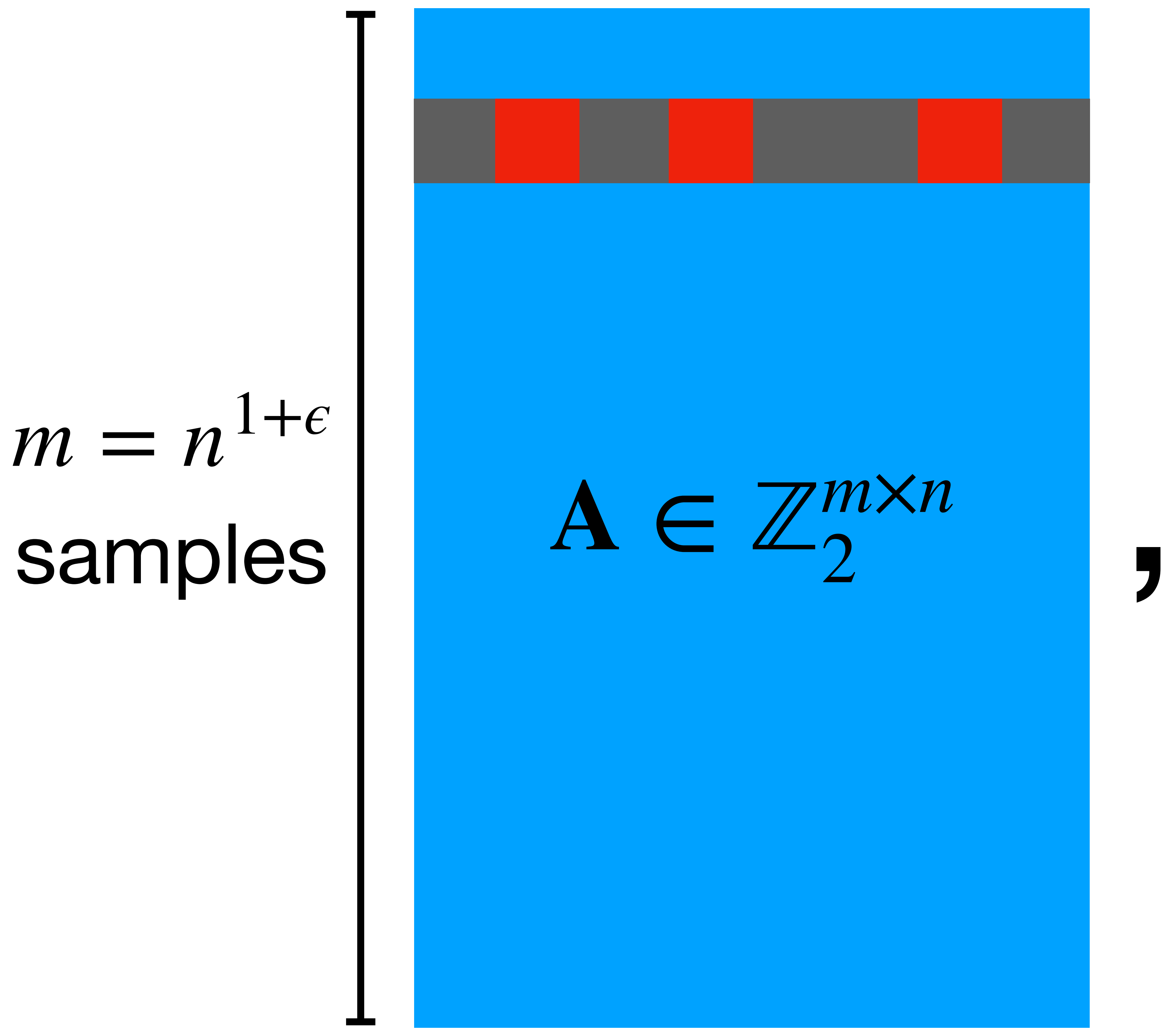
$$\mathbf{s} \leftarrow \mathbb{Z}_2^n$$
$$\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$$

Two ways to instantiate LFN:

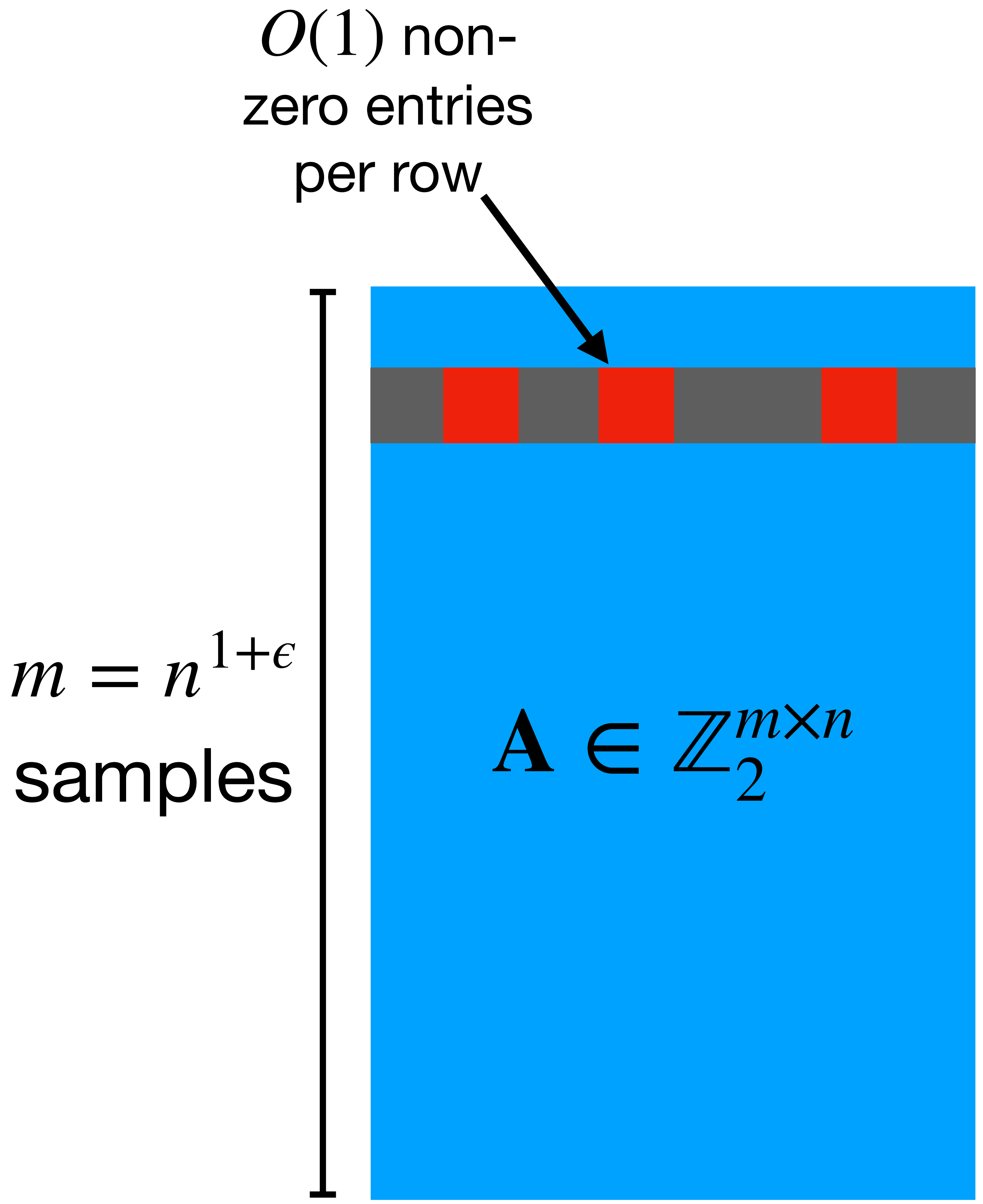
1. PRGs in NC^0 (e.g., Goldreich's PRGs): no noise! ($\delta \to \infty$).
2. Sparse LPN*: f is a $O(1)$-sparse, linear function over $\mathbb{Z}_2$.

*Generalizing Sparse LPN to LFN was suggested by Aayush Jain, Rachel Lin, and an anonymous reviewer.

Sparse LPN over $\mathbb{Z}_2$

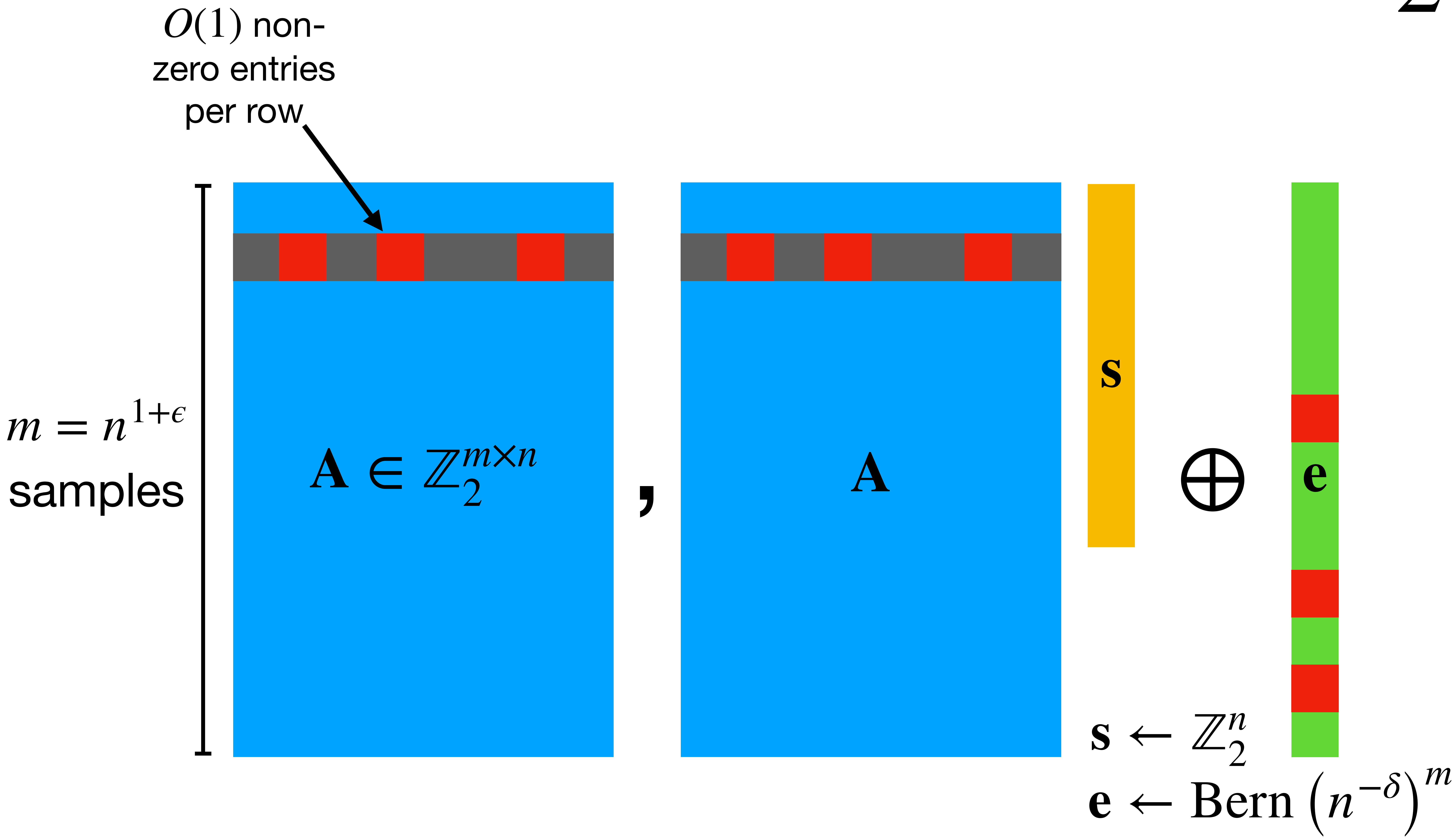


Sparse LPN over $\mathbb{Z}_2$

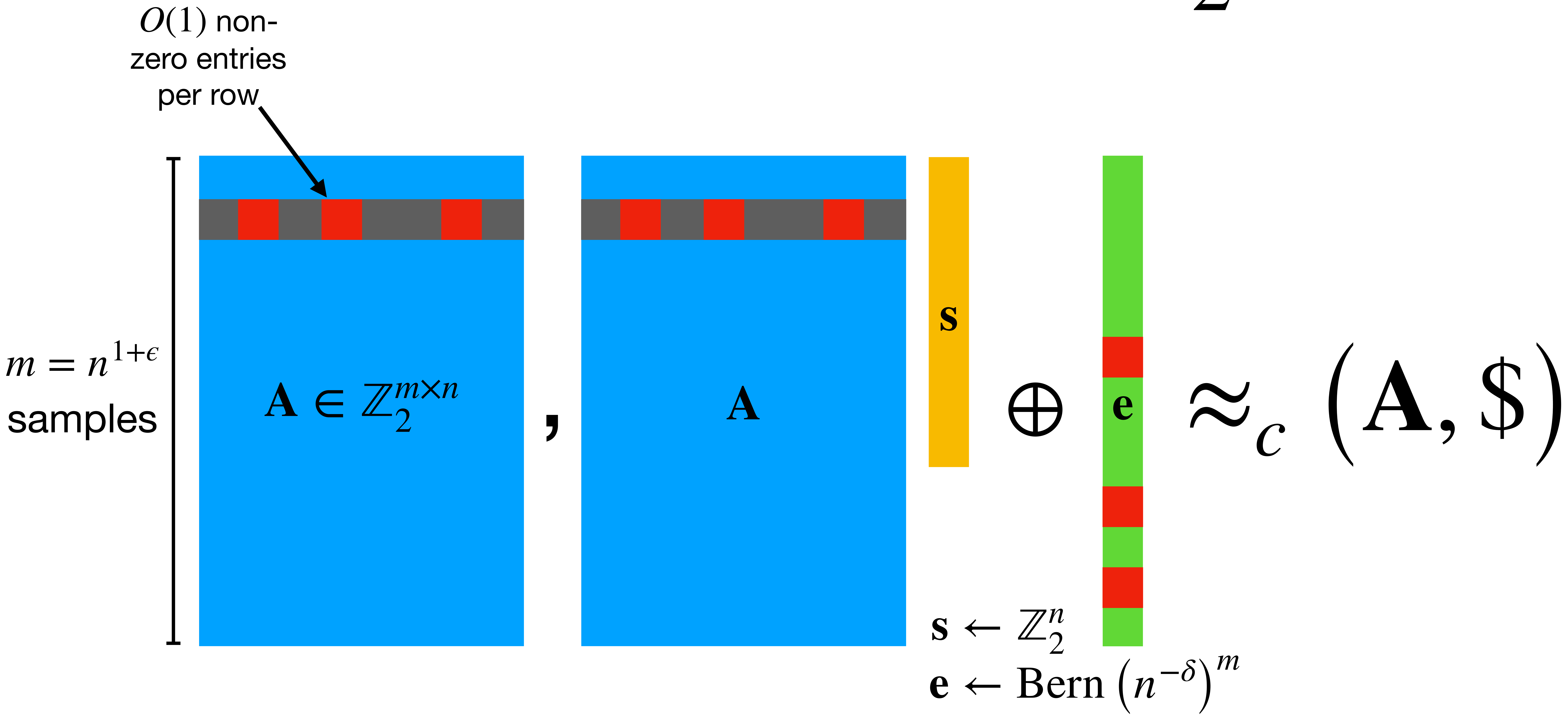


,

Sparse LPN over $\mathbb{Z}_2$



Sparse LPN over $\mathbb{Z}_2$



IO from “2.5 Assumptions”

Another interpretation:

IO from “2.5 Assumptions”

Another interpretation:

Theorem [R**V**V ’24]: Construction of IO from (sub-exponential)

- Bilinear Maps (DLIN) ,
- (Large-field) LPN, &
- **(Sparse) LPN**.

Overview of [JLS22]

Overview of [JLS22]

$\boxed{\text{Bilinear Maps}}$

$\downarrow$ [JLMS19] [Wee20] [GJLS21]

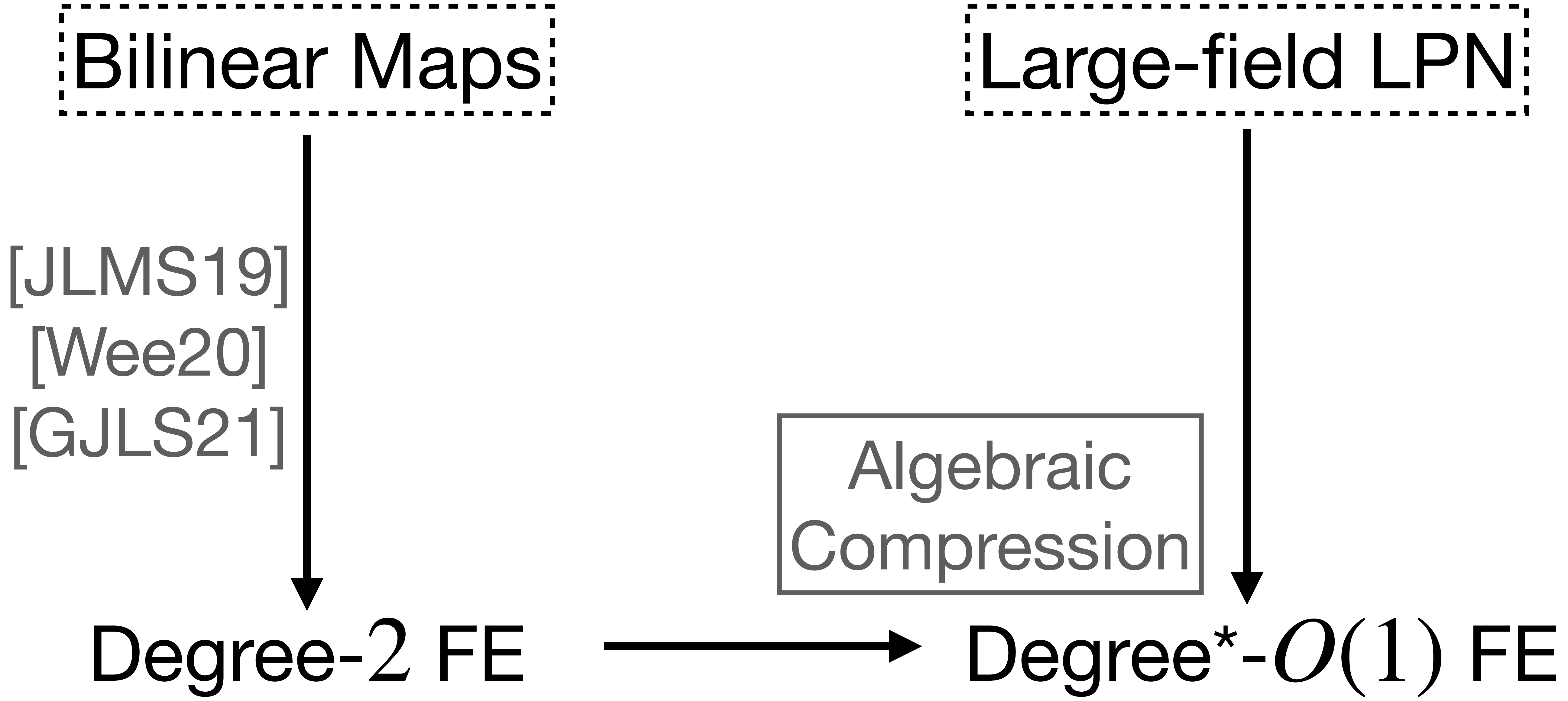
Degree-2 FE

Overview of [JLS22]

$$\begin{tikzcd} \boxed{\text{Bilinear Maps}} \arrow[d, "{\begin{array}{c}\text{[JLMS19]}\\ \text{[Wee20]}\\ \text{[GJLS21]}\end{array}}"'] & \boxed{\text{Large-field LPN}} \arrow[d] \\ \text{Degree-}2\text{ FE} \arrow[r] & \text{Degree*-}O(1)\text{ FE} \end{tikzcd}$$

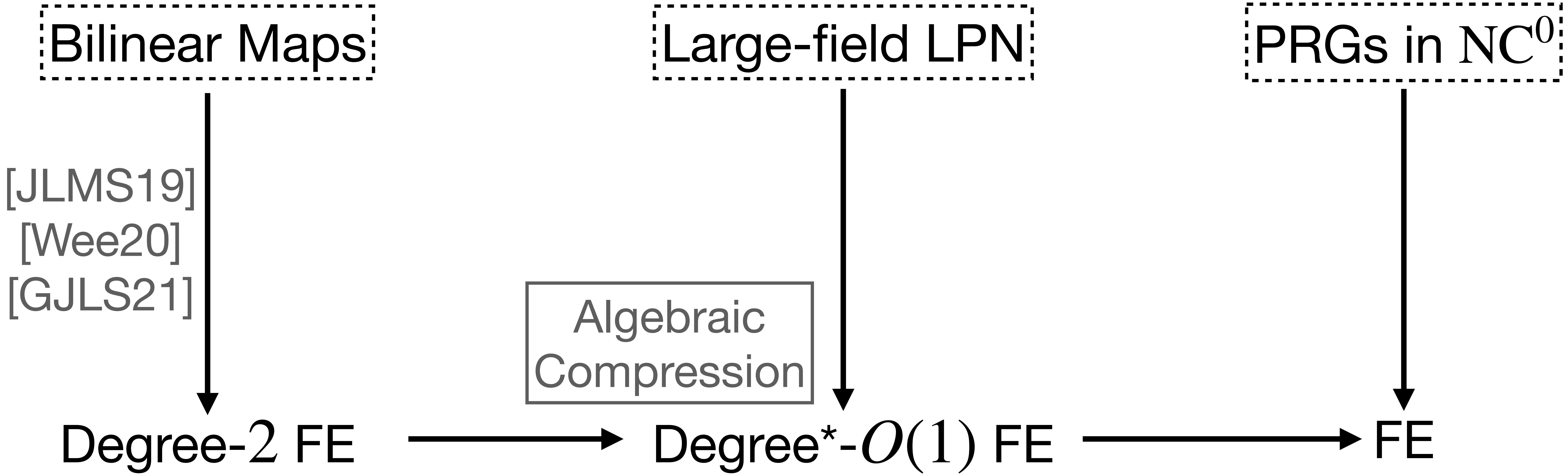
*Also need m^{ϵ} locality

Overview of [JLS22]



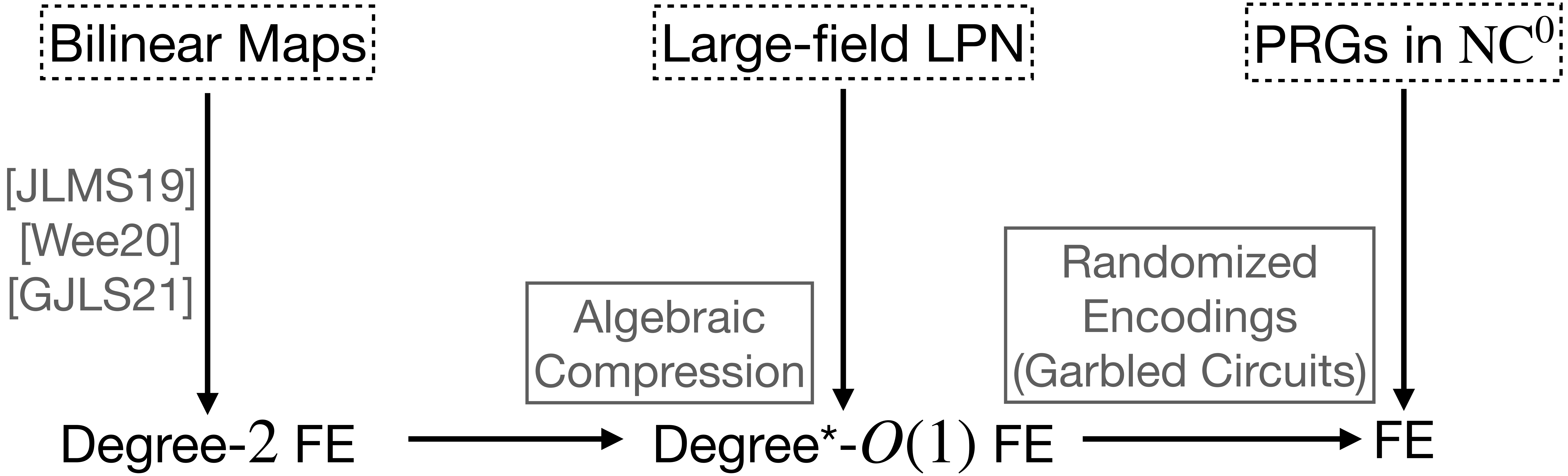
*Also need m^{ϵ} locality

Overview of [JLS22]



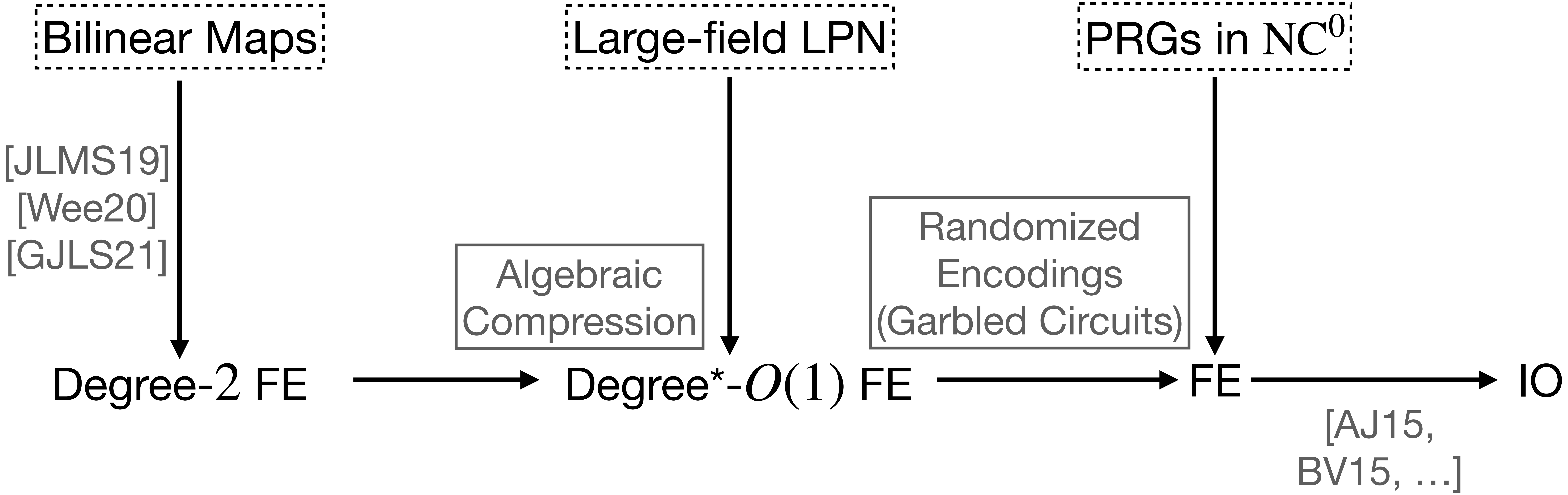
*Also need m^{ϵ} locality

Overview of [JLS22]



*Also need m^{ϵ} locality

Overview of [JLS22]



*Also need m^ϵ locality

Overview of [JLS22]

“Local Functions with Noise” (LFN)

Bilinear Maps

Large-field LPN

~~*PRGs in* NC^0~~

[JLMS19]
[Wee20]
[GJLS21]

Algebraic Compression

Randomized Encodings (Garbled Circuits)

$$\text{Degree-}2\text{ FE} \longrightarrow \text{Degree*-}O(1)\text{ FE} \longrightarrow \text{FE} \longrightarrow \text{IO}$$

[AJ15, BV15, …]

*Also need m^{ϵ} locality

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

1. **Linear**-stretch PRGs in NC^0.

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

1. **Linear**-stretch PRGs in NC^0. [AIK08] shows implied by LFN. ✓

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

1. **Linear**-stretch PRGs in NC^0. [AIK08] shows implied by LFN. ✓

2. **Structured-seed** PRG with **degree** $O(1)$ (over $\mathbb{Z}$) and **locality** m^ϵ:

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

1. **Linear**-stretch PRGs in NC^0. [AIK08] shows implied by LFN. ✓

2. **Structured-seed** PRG with **degree** $O(1)$ (over $\mathbb{Z}$) and **locality** m^{ϵ}:

   a) $\mathrm{SeedSample}\left(1^{\lambda}\right) \to \sigma$, where $\mathrm{SeedSample}$ takes $m^{1-\Omega(1)}$ time.

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

1. **Linear**-stretch PRGs in NC^0. [AIK08] shows implied by LFN.
2. **Structured-seed** PRG with **degree** $O(1)$ (over $\mathbb{Z}$) and **locality** m^ϵ:
   a) $\mathrm{SeedSample}\left(1^\lambda\right) \to \sigma$, where $\mathrm{SeedSample}$ takes $m^{1-\Omega(1)}$ time.
   b) $G : \{0,1\}^{m^{1-\Omega(1)}} \to \{0,1\}^m$ has degree $O(1)$ and locality m^ϵ.

Relaxing poly-stretch PRGs in NC^0

We observe two **weaker** objects suffice to build FE from Degree*-$O(1)$ FE:

1. **Linear**-stretch PRGs in NC^0. [AIK08] shows implied by LFN. ✓
2. **Structured-seed** PRG with **degree** $O(1)$ (over $\mathbb{Z}$) and **locality** m^{ϵ}:
   a) $\mathrm{SeedSample}\left(1^{\lambda}\right) \to \sigma$, where $\mathrm{SeedSample}$ takes $m^{1-\Omega(1)}$ time.
   b) $G : \{0,1\}^{m^{1-\Omega(1)}} \to \{0,1\}^{m}$ has degree $O(1)$ and locality m^{ϵ}.
   c) $G(\sigma) \approx_c \$$.

Structured-Seed PRG from LFN

First attempt:

Structured-Seed PRG from LFN

First attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m\right)$, where $m = n^{1+\epsilon}$.

Structured-Seed PRG from LFN

First attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m\right)$, where $m = n^{1+\epsilon}$.

2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$.

Structured-Seed PRG from LFN

First attempt:

1. SeedSample: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \mathbf{e} \leftarrow \text{Bern}\left(n^{-\delta}\right)^m\right)$, where $m = n^{1+\epsilon}$.

2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$.

What goes wrong?

Structured-Seed PRG from LFN

First attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m\right)$, where $m = n^{1+\epsilon}$.

2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$.

What goes wrong?

- If $\mathbf{e} \in \mathbb{Z}_2^m$ written in full, **not even expanding**.

Structured-Seed PRG from LFN

First attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m\right)$, where $m = n^{1+\epsilon}$.

2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$.

What goes wrong?

- If $\mathbf{e} \in \mathbb{Z}_2^m$ written in full, **not even expanding**.

- If $\mathbf{e}$ is written as list of non-zero indices, **expansion not in degree** $O(1)$.

Algebraic Compression

Algebraic Compression

Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Algebraic Compression

Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Yes! (in fact, degree 2) [JLS21, JLS22]

Algebraic Compression

Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Yes! (in fact, degree 2) [JLS21, JLS22]

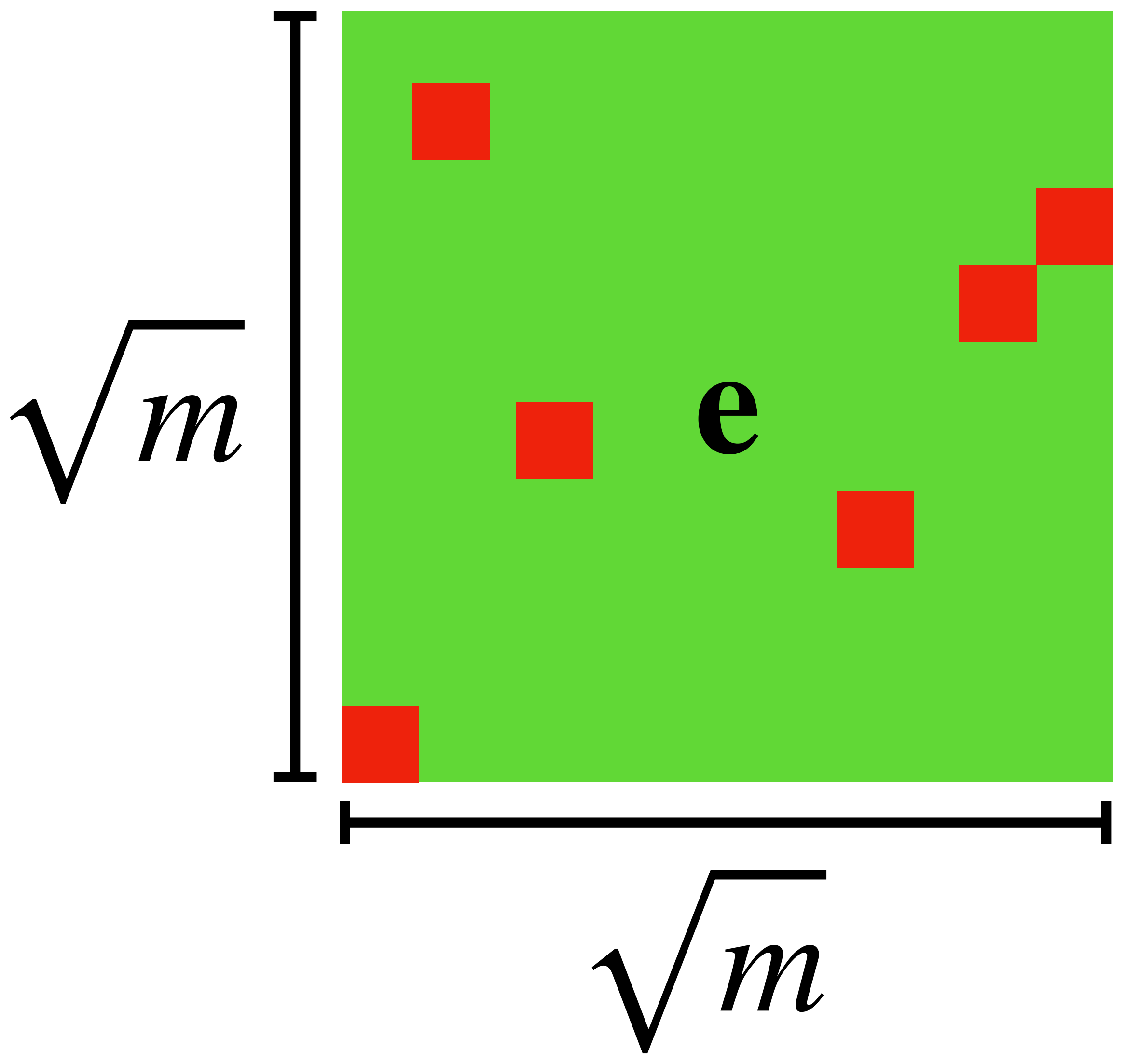
Key idea: Interpret $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ as a sparse, square **matrix**:

Algebraic Compression

Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Yes! (in fact, degree 2) [JLS21, JLS22]

Key idea: Interpret $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ as a sparse, square **matrix**:



Algebraic Compression

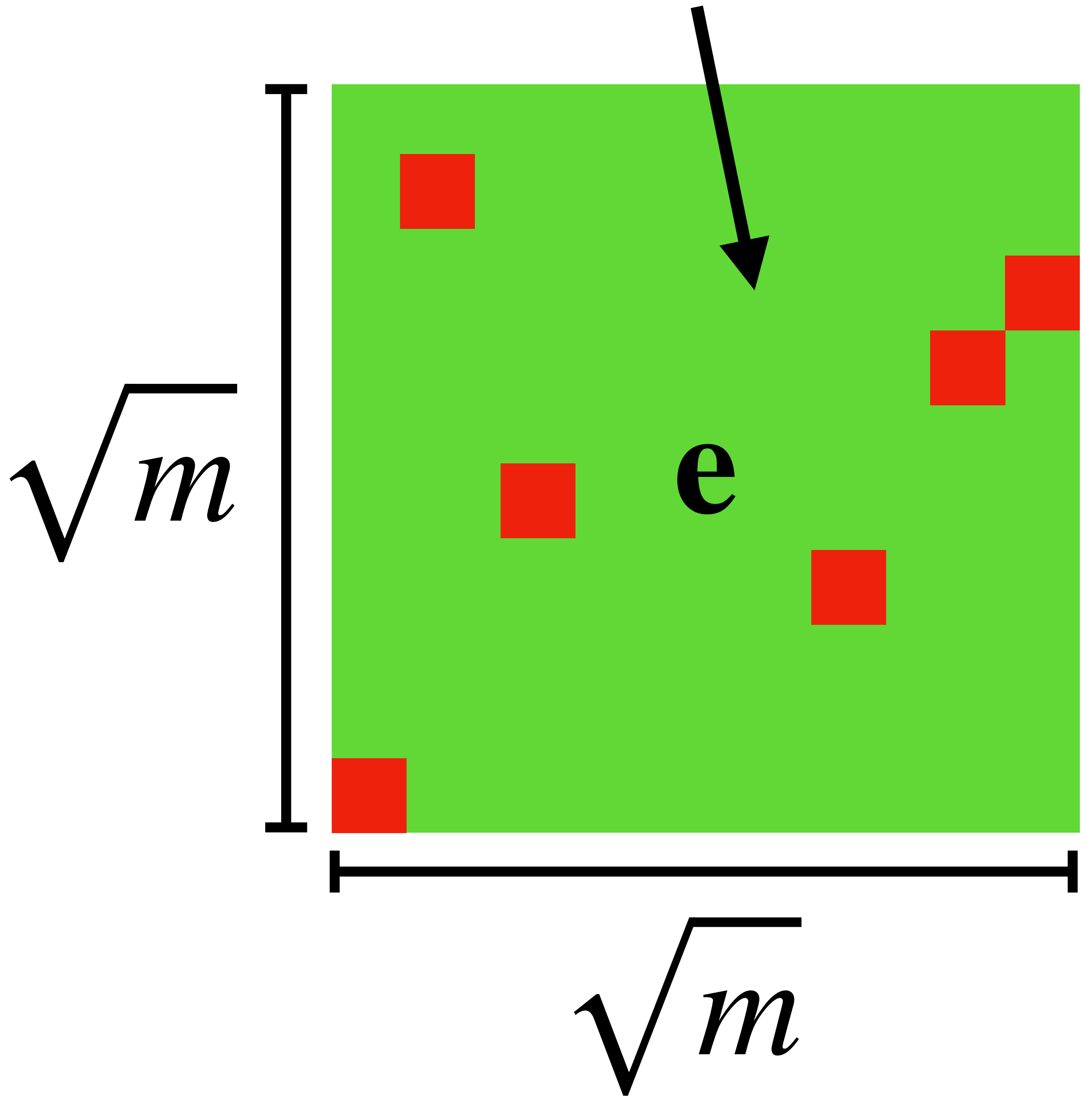
Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Yes! (in fact, degree 2) [JLS21, JLS22]

Key idea: Interpret $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ as a sparse, square **matrix**:

$mn^{-\delta} = m^{1/2-\Omega(1)}$ (can remove this assumption on δ)

non-zero entries

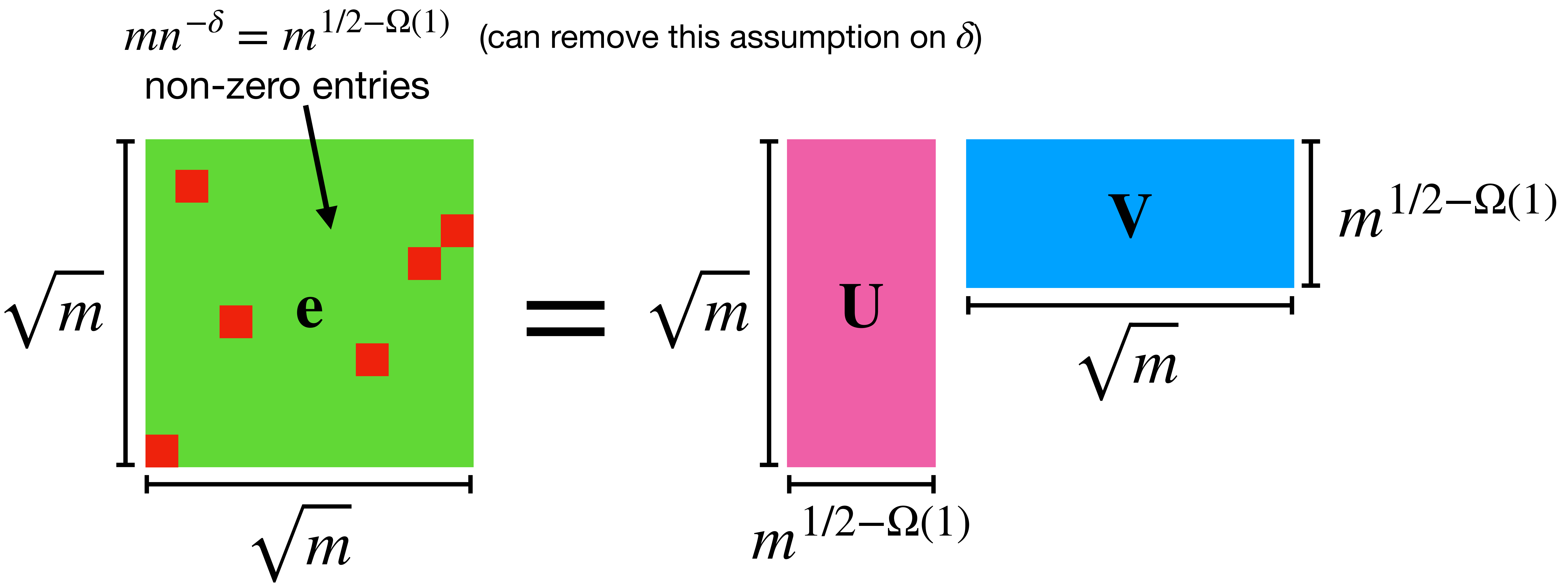


Algebraic Compression

Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Yes! (in fact, degree 2) [JLS21, JLS22]

Key idea: Interpret $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ as a sparse, square **matrix**:



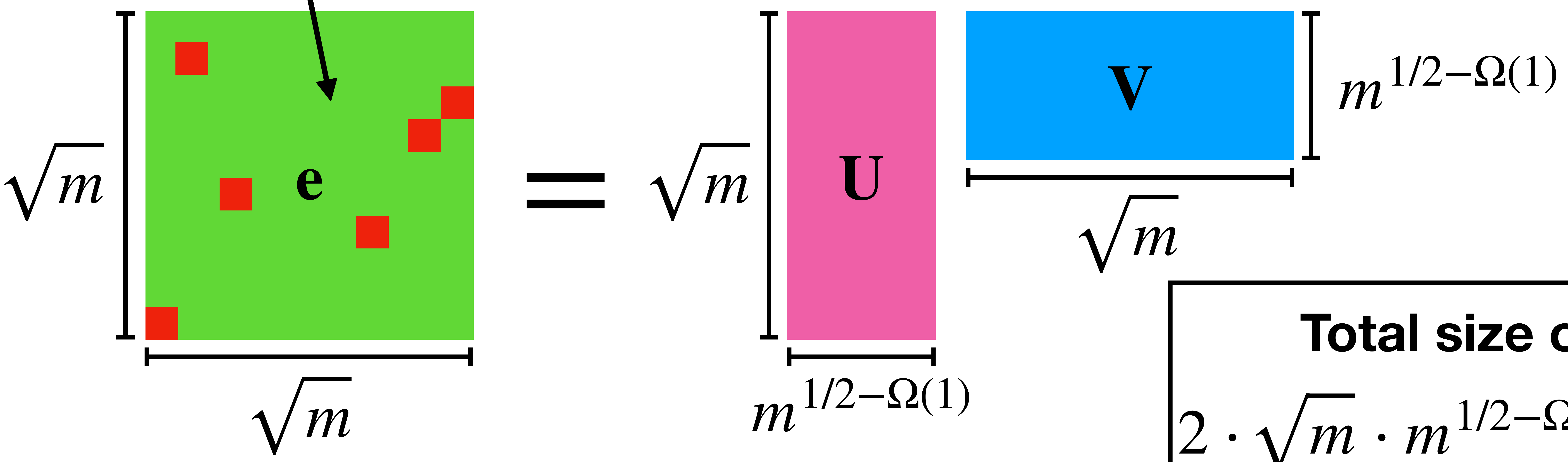
Algebraic Compression

Can we compress $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ so that expansion is degree $O(1)$?

Yes! (in fact, degree 2) [JLS21, JLS22]

Key idea: Interpret $\mathbf{e} \leftarrow \mathrm{Bern}\left(n^{-\delta}\right)^m$ as a sparse, square **matrix**:

$mn^{-\delta} = m^{1/2-\Omega(1)}$ (can remove this assumption on δ)
non-zero entries



Total size of $(\mathbf{U}, \mathbf{V})$:

$2 \cdot \sqrt{m} \cdot m^{1/2-\Omega(1)} = m^{1-\Omega(1)}$.

Structured-Seed PRG from LFN

Second attempt:

Structured-Seed PRG from LFN

Second attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \left(\mathbf{U}, \mathbf{V}^\top \in \mathbb{Z}_2^{\sqrt{m} \times m^{1/2-\Omega(1)}} \right) \right)$

Structured-Seed PRG from LFN

Second attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \left(\mathbf{U}, \mathbf{V}^\top \in \mathbb{Z}_2^{\sqrt{m} \times m^{1/2-\Omega(1)}}\right)\right)$
2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$, where $\mathbf{e}$ is a reshaping of $\mathbf{UV} \in \mathbb{Z}_2^{\sqrt{m}\times\sqrt{m}}$.

Structured-Seed PRG from LFN

Second attempt:

1. $\mathrm{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \left(\mathbf{U}, \mathbf{V}^\top \in \mathbb{Z}_2^{\sqrt{m} \times m^{1/2-\Omega(1)}}\right)\right)$

2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$, where $\mathbf{e}$ is a reshaping of $\mathbf{UV} \in \mathbb{Z}_2^{\sqrt{m}\times\sqrt{m}}$.

Check properties:

Structured-Seed PRG from LFN

Second attempt:

1. $\mathsf{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \left(\mathbf{U}, \mathbf{V}^\top \in \mathbb{Z}_2^{\sqrt{m} \times m^{1/2-\Omega(1)}}\right)\right)$

2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$, where $\mathbf{e}$ is a reshaping of $\mathbf{UV} \in \mathbb{Z}_2^{\sqrt{m}\times\sqrt{m}}$.

Check properties:

a) Seed σ can be computed in time $m^{1-\Omega(1)}$. ✓

Structured-Seed PRG from LFN

Second attempt:

1. $\mathsf{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \left(\mathbf{U}, \mathbf{V}^\top \in \mathbb{Z}_2^{\sqrt{m} \times m^{1/2-\Omega(1)}}\right)\right)$
2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$, where $\mathbf{e}$ is a reshaping of $\mathbf{UV} \in \mathbb{Z}_2^{\sqrt{m}\times\sqrt{m}}$.

Check properties:

a) Seed σ can be computed in time $m^{1-\Omega(1)}$. ✓

b) G_f has degree-$O(1)$ and locality $m^{1/2-\Omega(1)}$. ✓

Structured-Seed PRG from LFN

Second attempt:

1. $\mathsf{SeedSample}$: Output $\sigma = \left(\mathbf{s} \leftarrow \mathbb{Z}_2^n, \left(\mathbf{U}, \mathbf{V}^\top \in \mathbb{Z}_2^{\sqrt{m} \times m^{1/2-\Omega(1)}}\right)\right)$
2. $G_f(\sigma) = f(\mathbf{s}) \oplus \mathbf{e}$, where $\mathbf{e}$ is a reshaping of $\mathbf{UV} \in \mathbb{Z}_2^{\sqrt{m}\times\sqrt{m}}$.

Check properties:

a) Seed σ can be computed in time $m^{1-\Omega(1)}$. ✓

b) G_f has degree-$O(1)$ and locality $m^{1/2-\Omega(1)}$. ✓

c) Pseudorandom assuming LFN. ✓

Open Questions & Future Directions

Open Questions & Future Directions

1. Minimal assumptions for IO?

Open Questions & Future Directions

1. Minimal assumptions for IO?

2. Post-quantum IO? (Need to replace bilinear maps.)

Open Questions & Future Directions

1. Minimal assumptions for IO?

2. Post-quantum IO? (Need to replace bilinear maps.)

3. More crypto from LFN? Cryptanalysis?

Thanks!