

Multiparty Computation with Minimal Overhead without Circuit Transformation

CRYPTO 2026



Aditya Hegde

JHU



Mingyuan Wang

University of Delaware

Phuoc Van Long Pham

Brown University

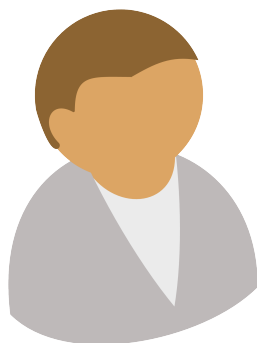
Can Secure Computation Achieve Ideal-World Efficiency?

Can Secure Computation Achieve Ideal-World Efficiency?

Ideal World

Arithmetic circuit C

x_1



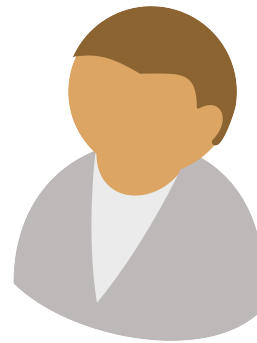
x_2



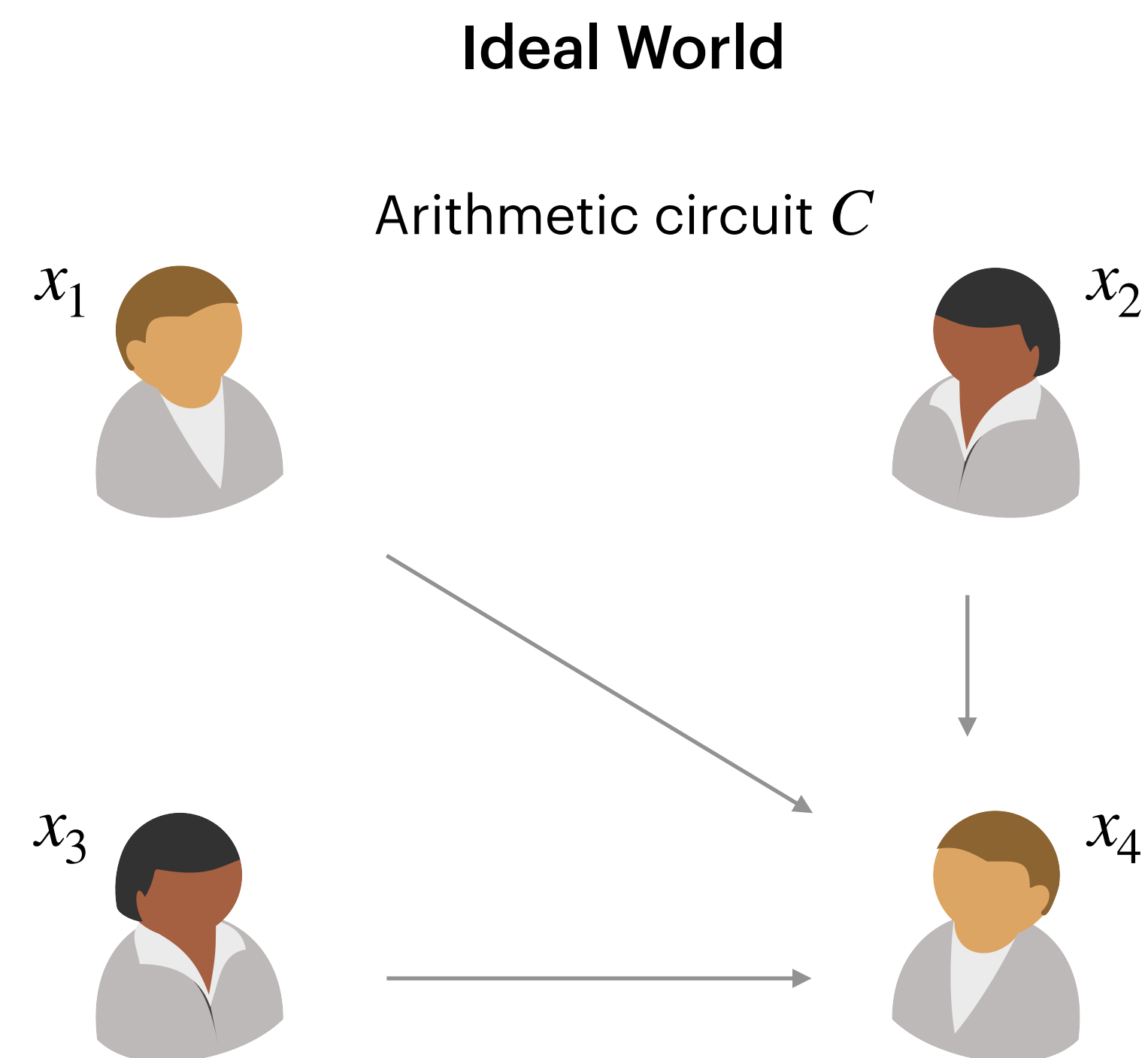
x_3



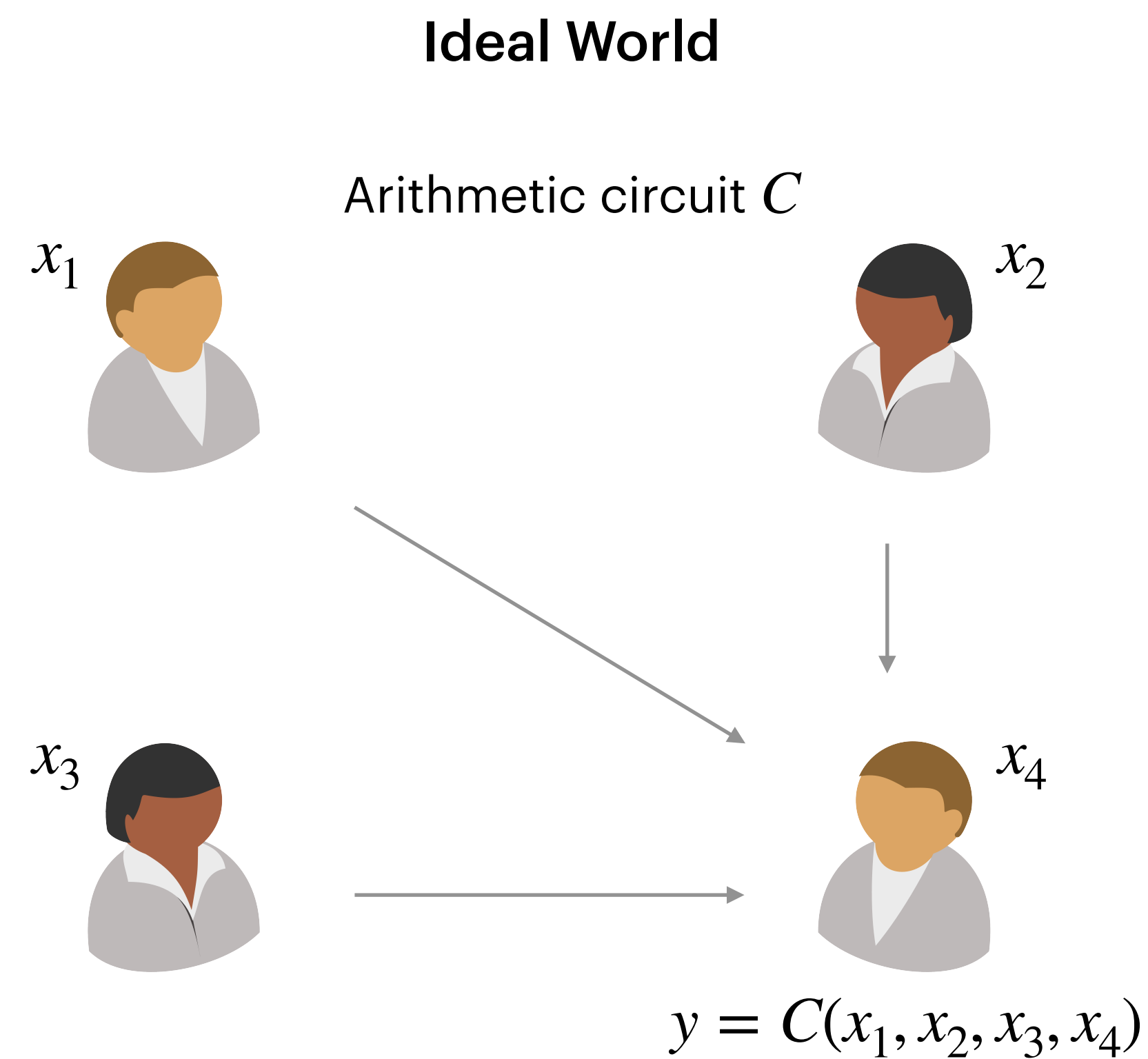
x_4



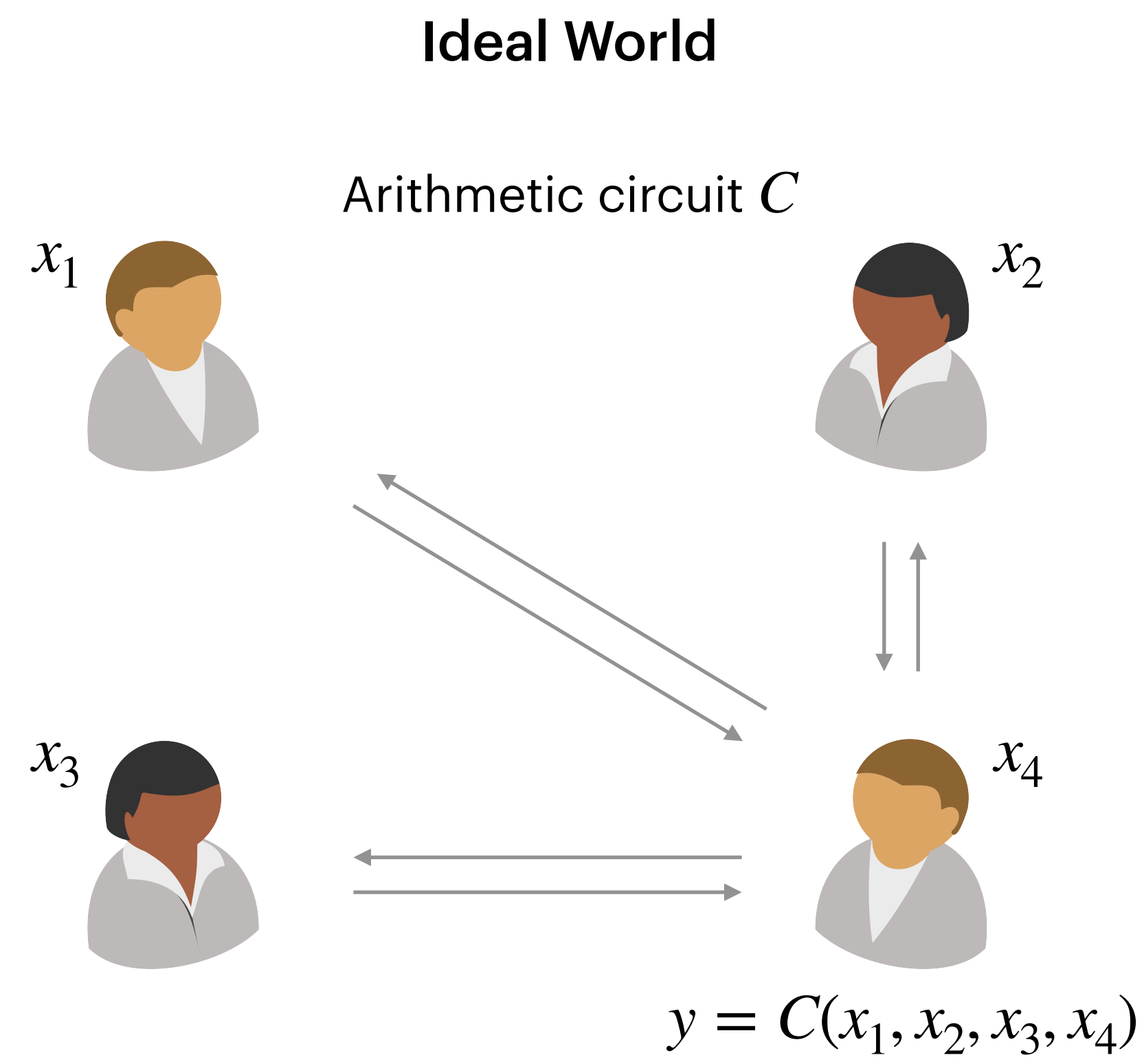
Can Secure Computation Achieve Ideal-World Efficiency?



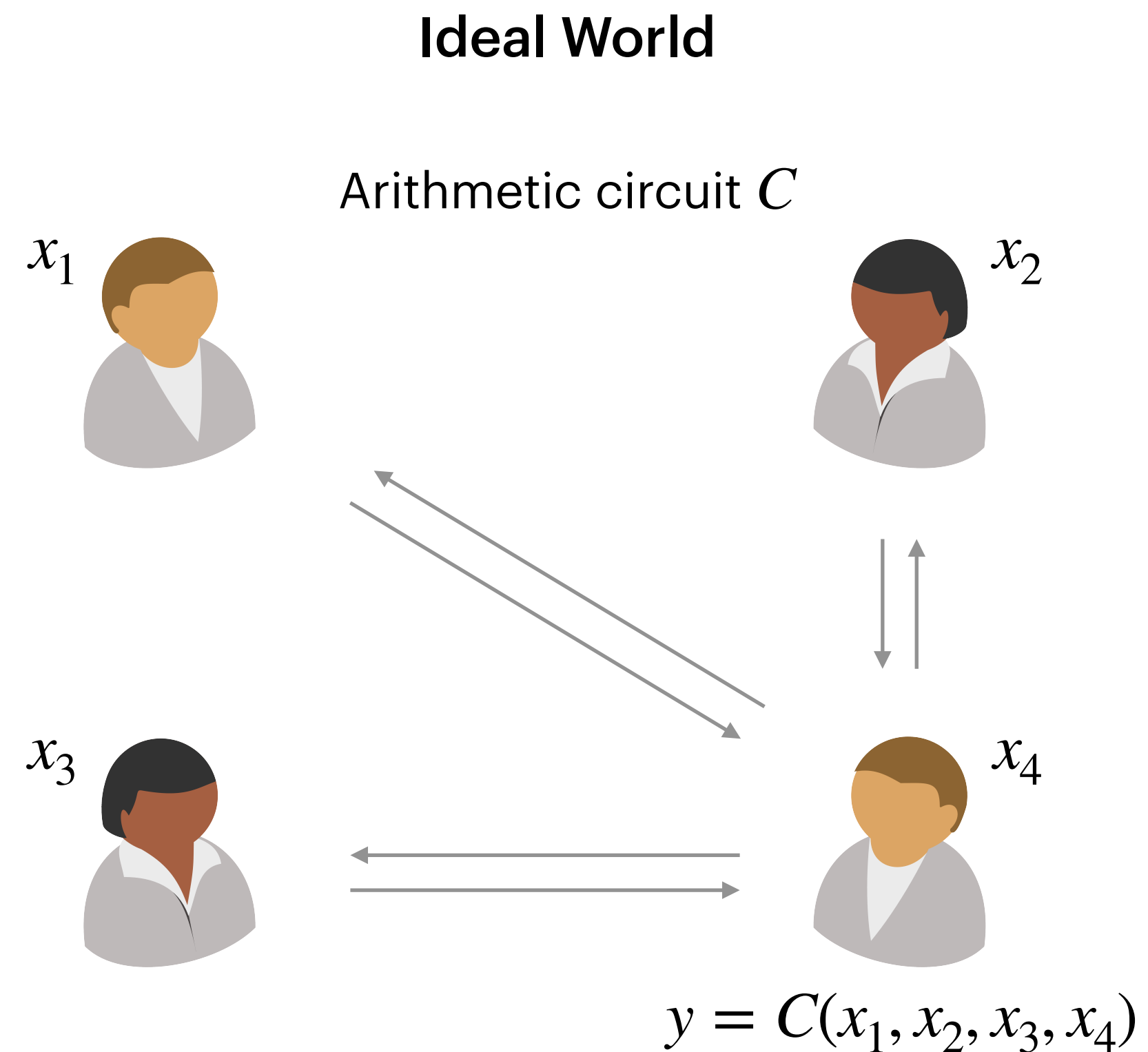
Can Secure Computation Achieve Ideal-World Efficiency?



Can Secure Computation Achieve Ideal-World Efficiency?



Can Secure Computation Achieve Ideal-World Efficiency?

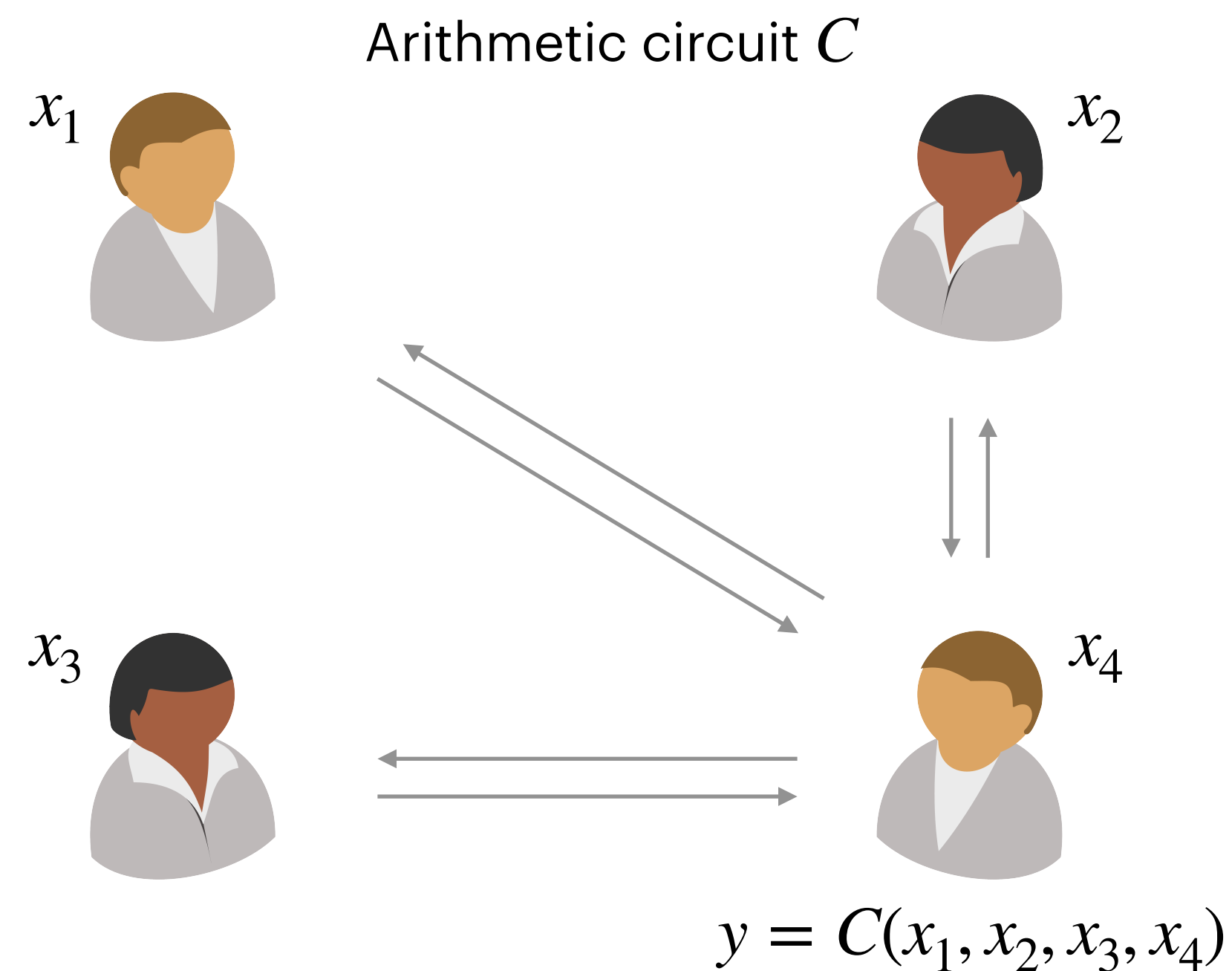


Total Computation

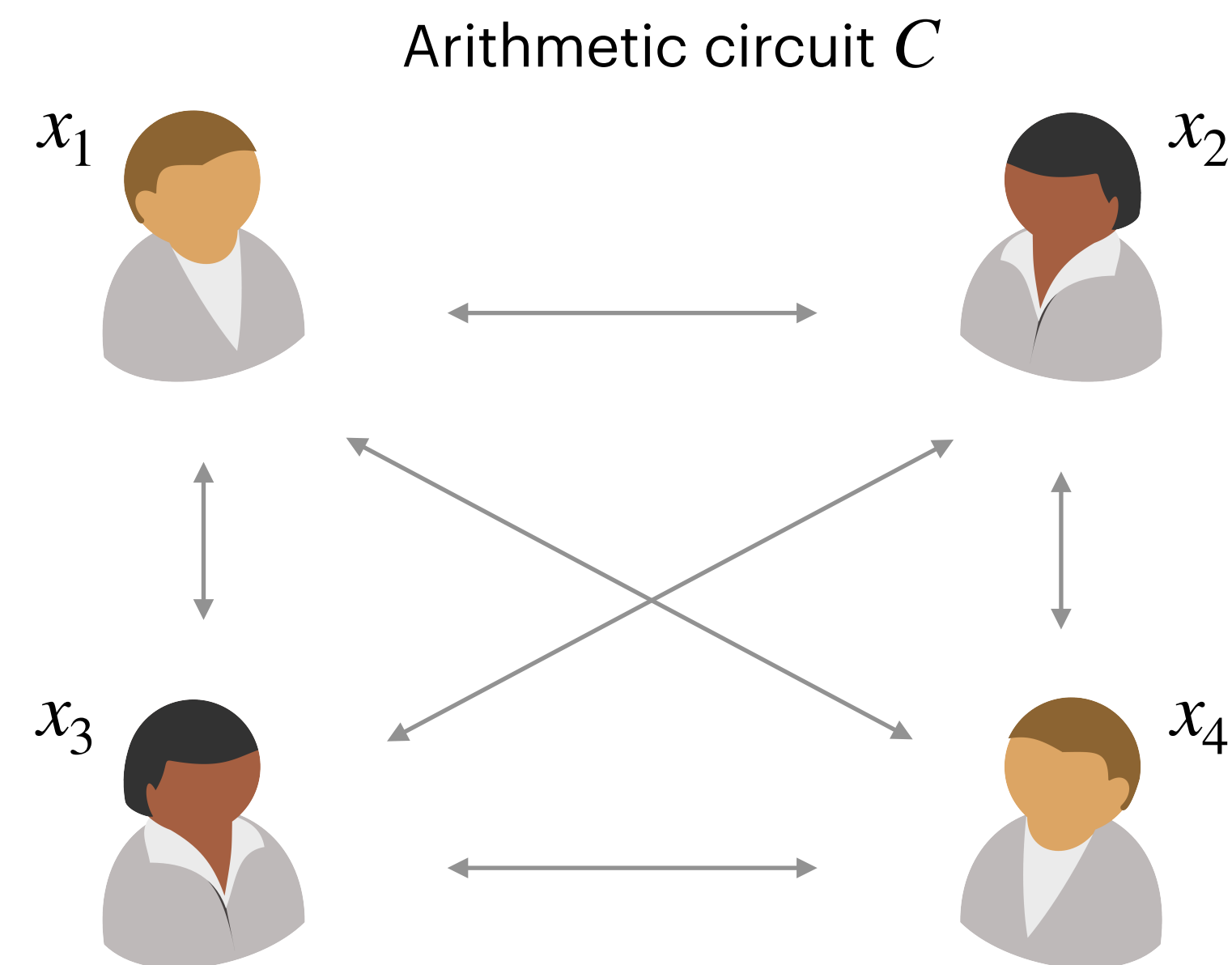
$$\sum_i \text{run-time}_i = |C| \text{ arithmetic operations}$$

Can Secure Computation Achieve Ideal-World Efficiency?

Ideal World



MPC

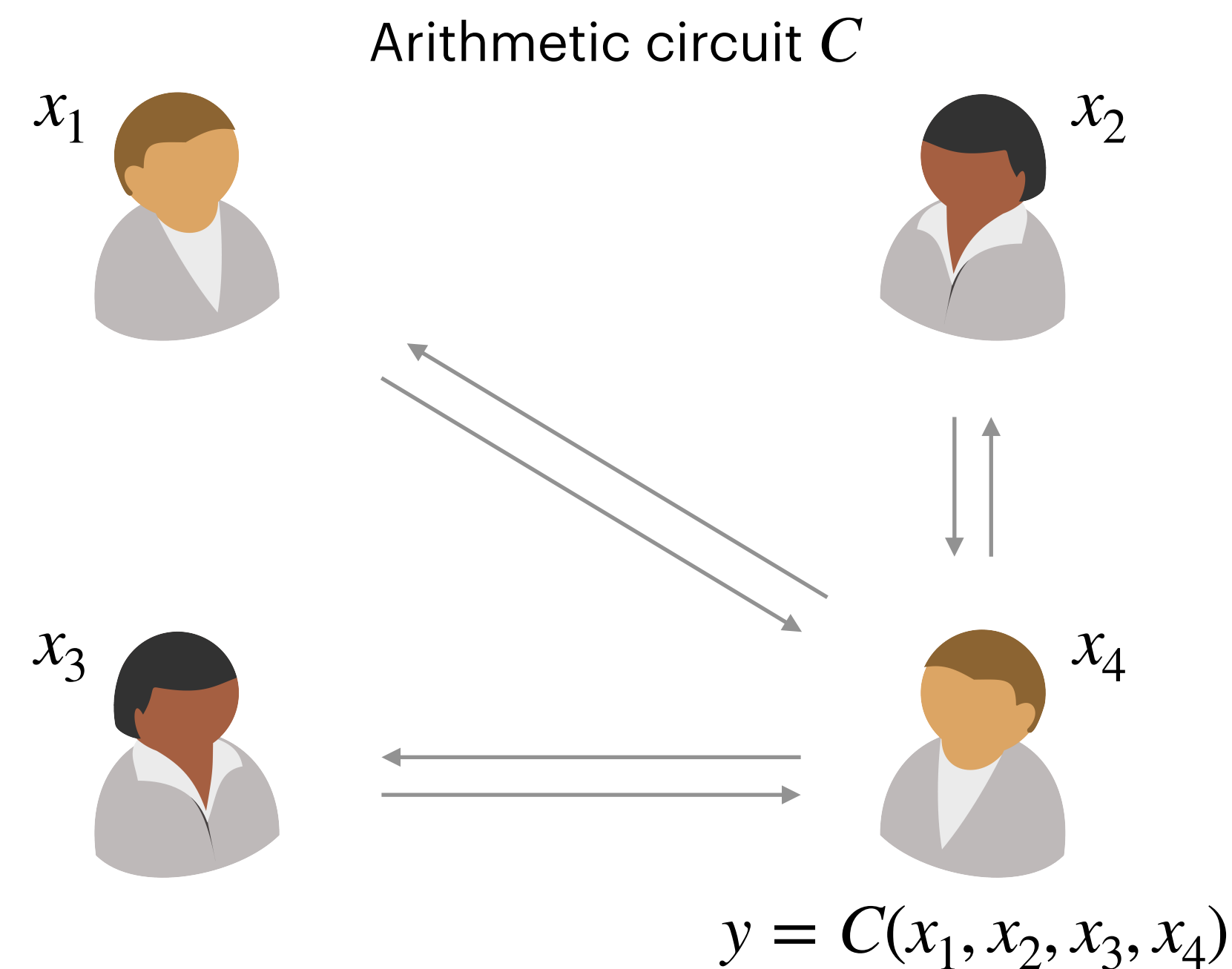


Total Computation

$$\sum_i \text{run-time}_i = |C| \text{ arithmetic operations}$$

Can Secure Computation Achieve Ideal-World Efficiency?

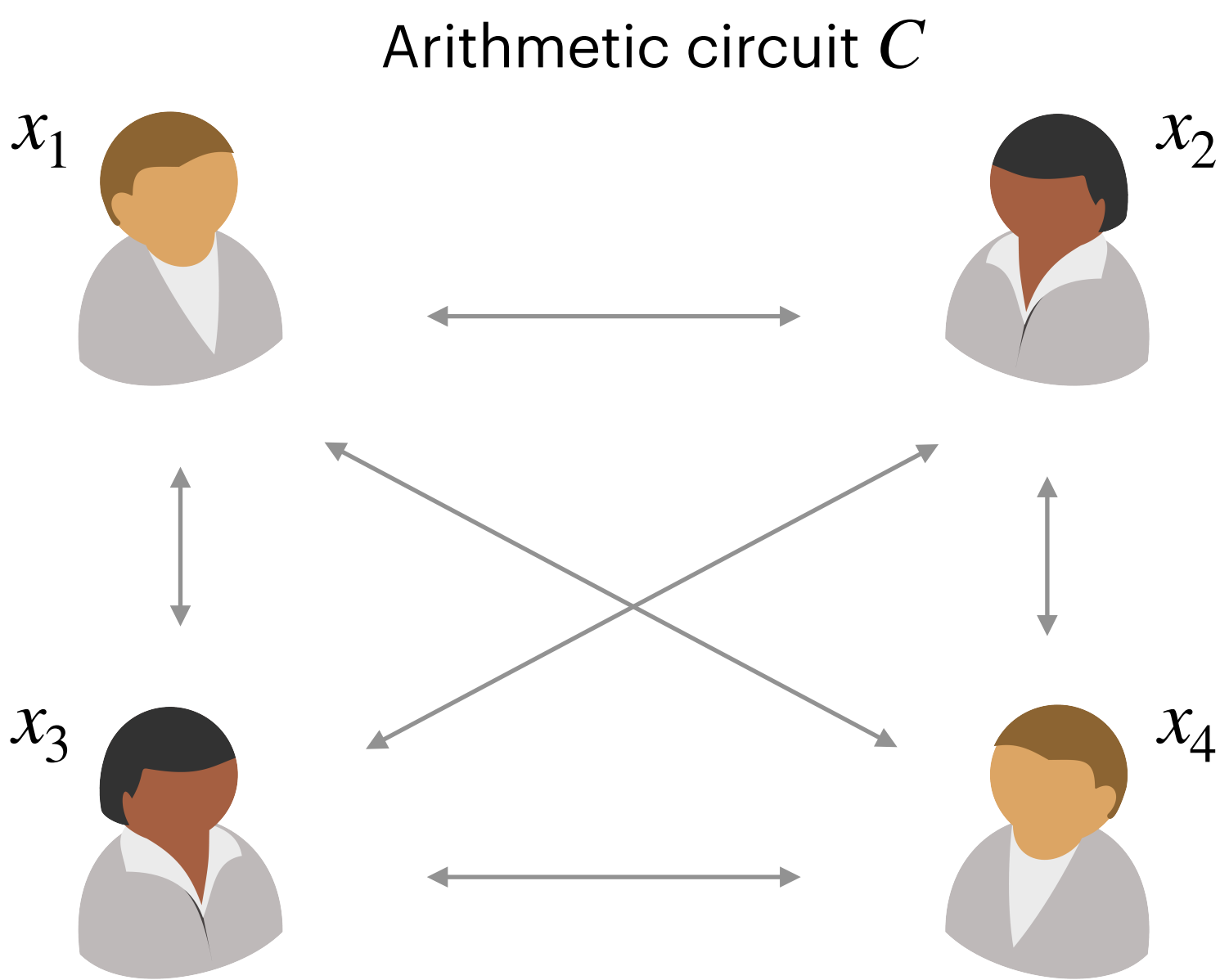
Ideal World



Total Computation

$$\sum_i \text{run-time}_i = |C| \text{ arithmetic operations}$$

MPC

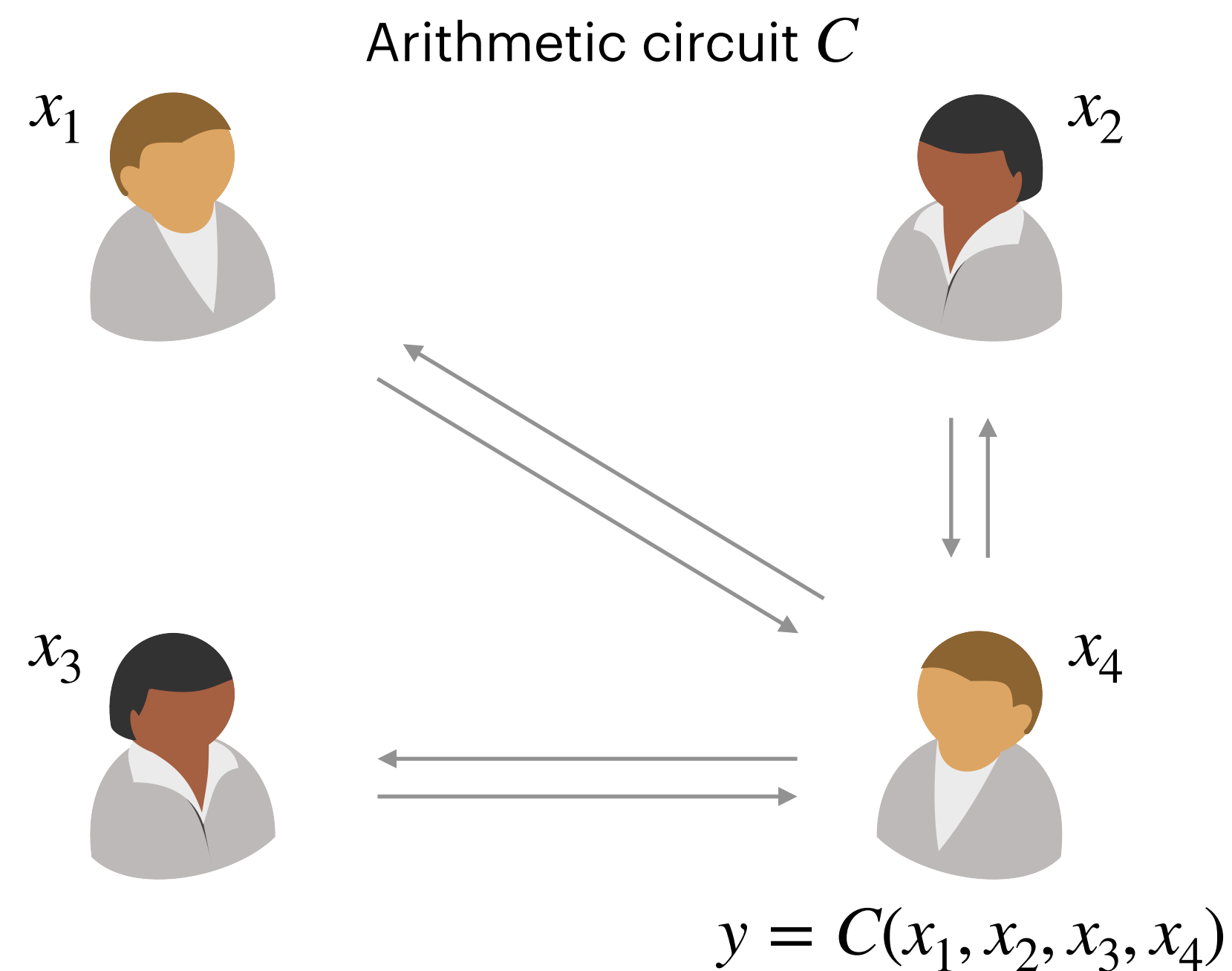


MPC with Minimal Overhead

$$\sum_i \text{run-time}_i = O(|C|) \text{ arithmetic operations}$$

Can Secure Computation Achieve Ideal-World Efficiency?

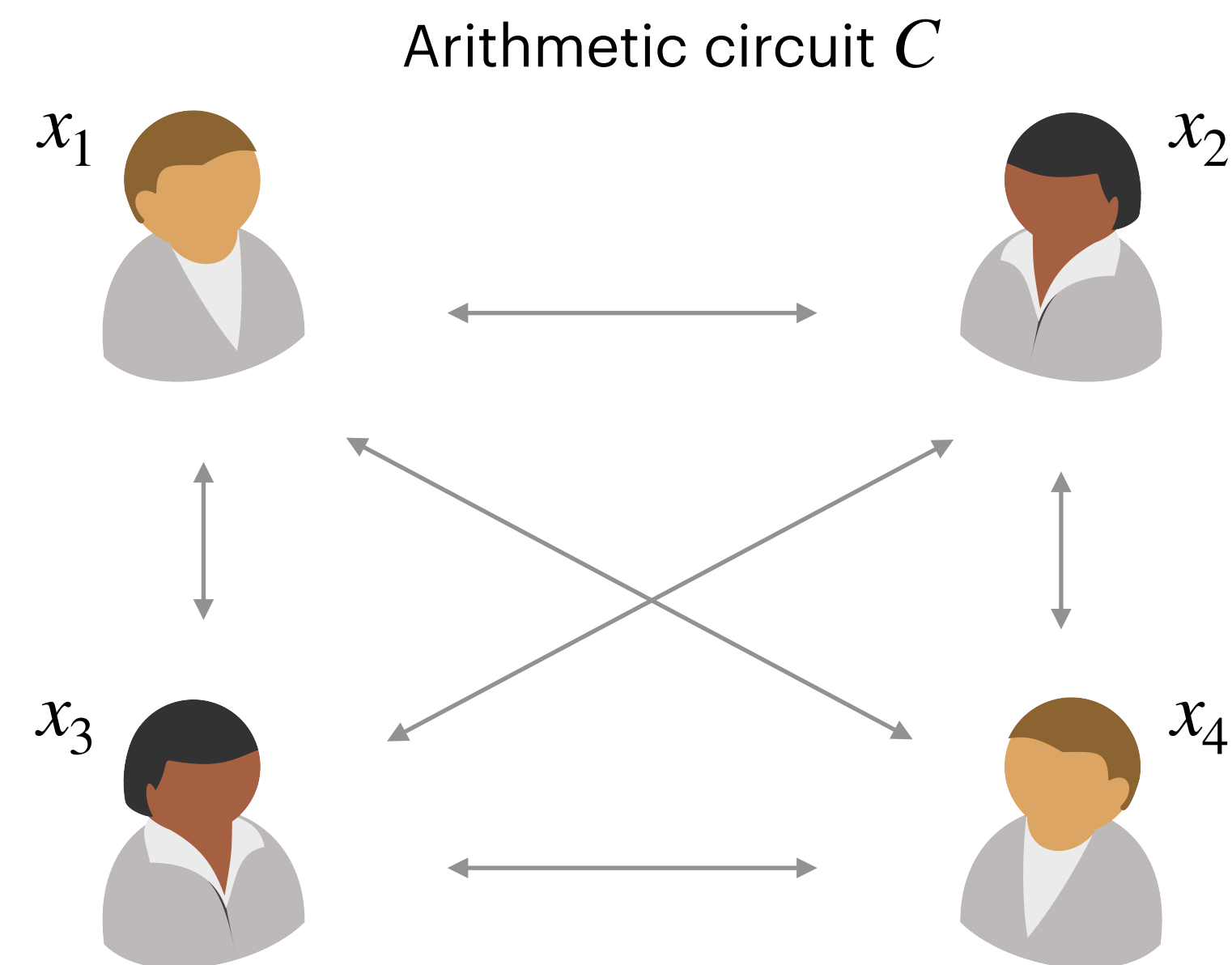
Ideal World



Total Computation

$$\sum_i \text{run-time}_i = |C| \text{ arithmetic operations}$$

MPC



Suppresses polylog factors
in number of parties

with Minimal Overhead

$$\sum_i \text{run-time}_i = O(|C|) \text{ arithmetic operations}$$

Prior Work

- Known in information-theoretic, honest-majority setting

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits

[Beck–Goel–Jain–Kaptchuk’21]

Highly repetitive circuits (e.g., SIMD)

$O(|C|)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits

[Beck–Goel–Jain–Kaptchuk’21]

Highly repetitive circuits (e.g., SIMD)

$O(|C|)$ ops

Circuit Transformation

Folklore [Beck–Goel–Jain–Kaptchuk’21]

Circuits over large rings

$O(|C|)$ ops

[Damgård–Ishai–Krøigaard’10]

Any Boolean circuit

$O(|C| \cdot \log |C|)$ ops

Circuit transformation induces large concrete overhead

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops

Circuit transformation induces large concrete overhead

Sequence of recent works on MPC **without circuit transformation** and $O(|C|)$ ring element communication
[Goyal–Polychroniadou–Song’21] [Goyal–Polychroniadou–Song’22]
[Escudero–Goyal–Polychroniadou–Song’22] [Escudero–Goyal–Polychroniadou–Song–Weng’23]

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops

Circuit transformation induces large concrete overhead

Sequence of recent works on MPC **without circuit transformation** and $O(|C|)$ ring element communication
[Goyal–Polychroniadou–Song’21] [Goyal–Polychroniadou–Song’22]

Concretely efficient! [Escudero–Goyal–Polychroniadou–Song’22] [Escudero–Goyal–Polychroniadou–Song–Weng’23]

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops

Circuit transformation induces large concrete overhead

Sequence of recent works on MPC **without circuit transformation** and $O(|C|)$ ring element communication

[Goyal–Polychroniadou–Song’21] [Goyal–Polychroniadou–Song’22]

Concretely efficient! [Escudero–Goyal–Polychroniadou–Song’22] [Escudero–Goyal–Polychroniadou–Song–Weng’23]

Total computational complexity of these works is $O(n \cdot |C|)$

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits

[Beck–Goel–Jain–Kaptchuk’21]

Highly repetitive circuits (e.g., SIMD)

$O(|C|)$ ops

Circuit Transformation

Folklore [Beck–Goel–Jain–Kaptchuk’21]

Circuits over large rings

$O(|C|)$ ops

[Damgård–Ishai–Krøigaard’10]

Any Boolean circuit

$O(|C| \cdot \log |C|)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits

[Beck–Goel–Jain–Kaptchuk’21]

Highly repetitive circuits (e.g., SIMD)

$O(|C|)$ ops

Circuit Transformation

Folklore [Beck–Goel–Jain–Kaptchuk’21]

Circuits over large rings

$O(|C|)$ ops

[Damgård–Ishai–Krøigaard’10]

Any Boolean circuit

$O(|C| \cdot \log |C|)$ ops

Without Circuit Transformation

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops
Without Circuit Transformation		
[Cramer–Damgård–de-Haan’07]	Circuits over large field extensions	$O(C)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops
Without Circuit Transformation		
[Cramer–Damgård–de-Haan’07]	Circuits over large field extensions	$O(C)$ ops
[Garg–Jain–Mukherjee–Wang’24]	Circuits over large integer rings	$O(C)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck–Goel–Jain–Kaptchuk’21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck–Goel–Jain–Kaptchuk’21]	Circuits over large rings	$O(C)$ ops
[Damgård–Ishai–Krøigaard’10]	Any Boolean circuit	$O(C \cdot \log C)$ ops
Without Circuit Transformation		
[Cramer–Damgård–de-Haan’07]	Circuits over large field extensions	$O(C)$ ops
[Garg–Jain–Mukherjee–Wang’24]	Circuits over large integer rings	$O(C)$ ops

Prior Work

- Known in information-theoretic, honest-majority setting

Structured Circuits		
[Beck-Goel-Jain-Kaptchuk'21]	Highly repetitive circuits (e.g., SIMD)	$O(C)$ ops
Circuit Transformation		
Folklore [Beck-Goel-Jain-Kaptchuk'21]	Circuits over large rings	$O(C)$ ops
[Damgård-Ishai-Krøigaard'10]	Any Boolean circuit	$O(C \cdot \log C)$ ops
Without Circuit Transformation		
[Cramer-Damgård-de-Haan'07]	Circuits over large field extensions	$O(C)$ ops
GJMW24	Circuits over large integer rings	$O(C)$ ops

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is co-prime to $\{2, \dots, n-1\}$

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Why can we hope to do better?

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Why can we hope to do better?

Key technical tool used by GJMW24 is **CRT-based secret sharing**

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Why can we hope to do better?

Key technical tool used by GJMW24 is **CRT-based secret sharing**



Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Why can we hope to do better?

Key technical tool used by GJMW24 is **CRT-based secret sharing**



Minimal overhead for secret sharing: $\text{secret-size} = \Omega\left(\sum_i \text{share-size}_i\right)$

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Why can we hope to do better?

Key technical tool used by GJMW24 is **CRT-based secret sharing**



Minimal overhead for secret sharing: $\log m = \Omega\left(\sum_i \log p_i\right)$

Limitations of Prior Work

GJMW24

n -party MPC protocol for computing over $\mathbb{Z}_m$ with total computation complexity $O(|C|)$ ring operations when

1. $\log m = \Omega(n^2 \log n)$
2. m is **co-prime** to $\{2, \dots, n-1\}$ e.g., this excludes $\mathbb{Z}_{2^k}$

Why can we hope to do better?

Key technical tool used by GJMW24 is **CRT-based secret sharing**



Minimal overhead for secret sharing: $\log m = \Omega\left(\sum_i \log p_i\right) = \Omega(n \log n)$

Our Result

n -party MPC protocol for evaluating any arithmetic circuit C over $\mathbb{Z}_m$ such that

total computation is $O(|C| \cdot \log m)$ when $\log m = \Omega(n \log n)$.

Our Result

n -party MPC protocol for evaluating any arithmetic circuit C over $\mathbb{Z}_m$ such that

total computation is $O(|C| \cdot \log m)$ when $\log m = \Omega(n \log n)$.

Measured in bit-cost RAM
model

Our Result

n -party MPC protocol for evaluating any arithmetic circuit C over $\mathbb{Z}_m$ such that

total computation is $O(|C| \cdot \log m)$ when $\log m = \Omega(n \log n)$.

- Information-theoretic security against adaptive semi-honest adversary

Our Result

n -party MPC protocol for evaluating any arithmetic circuit C over $\mathbb{Z}_m$ such that

total computation is $O(|C| \cdot \log m)$ when $\log m = \Omega(n \log n)$.

- Information-theoretic security against adaptive semi-honest adversary
- Honest majority with **suboptimal corruption threshold** i.e., for $0 < \epsilon < 1/2$

$$\frac{t}{n} < \frac{1}{2} - \epsilon$$

Our Result

n -party MPC protocol for evaluating any arithmetic circuit C over $\mathbb{Z}_m$ such that

total computation is $O(|C| \cdot \log m)$ when $\log m = \Omega(n \log n)$.

- Information-theoretic security against adaptive semi-honest adversary
- Honest majority with **suboptimal corruption threshold** i.e., for $0 < \epsilon < 1/2$

$$\frac{t}{n} < \frac{1}{2} - \epsilon$$

Necessary for secure computation with
minimal overhead

[Damgård–Larsen–Nielsen'19]

Our Result

n -party MPC protocol for evaluating any arithmetic circuit C over $\mathbb{Z}_m$ such that

total computation is $O(|C| \cdot \log m)$ when $\log m = \Omega(n \log n)$.

- Information-theoretic security against adaptive semi-honest adversary
- Honest majority with **suboptimal corruption threshold** i.e., for $0 < \epsilon < 1/2$

$$\frac{t}{n} < \frac{1}{2} - \epsilon$$

Necessary for secure computation with
minimal overhead
[Damgård-Larsen-Nielsen'19]

Overcomes both limitations of GJM24!

Overview of GJMW24

Overview of GJM24

- CRT-based Secret Sharing: Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$
- **Multiplication:** $[x] \cdot [y] = [XY]$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$
- **Multiplication:** $[x] \cdot [y] = [XY]$

XY should be reduced
modulo m

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$
- **Multiplication:** $[x] \cdot [y] = [XY]$
 - Reconstruct $[XY] + [R]$

XY should be reduced modulo m

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$
- **Multiplication:** $[x] \cdot [y] = [XY]$
 - Reconstruct $[XY] + [R]$
 - Reduce $xy + r = (XY + R) \bmod m$

XY should be reduced modulo m

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$

XY should be reduced modulo m
- **Multiplication:** $[x] \cdot [y] = [XY]$
 - Reconstruct $[XY] + [R]$
 - Reduce $xy + r = (XY + R) \bmod m$
 - Compute $[xy] = xy + r - [r]$

$r = R \bmod m$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$
 - XY should be reduced modulo m
- **Multiplication:** $[x] \cdot [y] = [XY]$
 - Reconstruct $[XY] + [R]$
 - Reconstructed over integers
 - Reduce $xy + r = (XY + R) \bmod m$
 - Compute $[xy] = xy + r - [r]$
 - $r = R \bmod m$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$

XY should be reduced modulo m
- **Multiplication:** $[x] \cdot [y] = [XY]$

Reconstructed over integers

 - Reconstruct $[XY] + [R]$
 - Reduce $xy + r = (XY + R) \bmod m$
 - Compute $[xy] = xy + r - [r]$

$r = R \bmod m$
- **Preprocessing:** Shares of integer masks $([R], [r])$ for every multiplication gate

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$

- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$

- **Addition:** $[x] + [y] = [x + y]$

XY should be reduced modulo m

- **Multiplication:** $[x] \cdot [y] = [XY]$

- Reconstruct $[XY] + [R]$

Reconstructed over integers

- Reduce $xy + r = (XY + R) \bmod m$

- Compute $[xy] = xy + r - [r]$

$r = R \bmod m$

- **Preprocessing:** Shares of integer masks $([R], [r])$ for every multiplication gate

Total computation is $O(|C| \log m)$,
assuming preprocessing, if

$$\log m = \Omega\left(\sum_i \log p_i\right) = \Omega(\log P_{\text{all}})$$

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$
- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$
- **Addition:** $[x] + [y] = [x + y]$
 - XY should be reduced modulo m
- **Multiplication:** $[x] \cdot [y] = [XY]$
 - Reconstruct $[XY] + [R]$
 - Reconstructed over integers
 - Reduce $xy + r = (XY + R) \bmod m$
 - Compute $[xy] = xy + r - [r]$
 - $r = R \bmod m$
- **Preprocessing:** Shares of integer masks $([R], [r])$ for every multiplication gate

Overview of GJM24

- **CRT-based Secret Sharing:** Share $(s \in \mathbb{Z}_m) \rightarrow [s] = ([s]_1 \in \mathbb{Z}_{p_1}, \dots, [s]_n \in \mathbb{Z}_{p_n})$

- Shares are homomorphic for P_{all} -bounded integer computation, where $P_{\text{all}} = \prod_i p_i$

- **Addition:** $[x] + [y] = [x + y]$

XY should be reduced modulo m

- **Multiplication:** $[x] \cdot [y] = [XY]$

- Reconstruct $[XY] + [R]$

Reconstructed over integers

- Reduce $xy + r = (XY + R) \bmod m$

- Compute $[xy] = xy + r - [r]$

$r = R \bmod m$

- **Preprocessing:** Shares of integer masks $([R], [r])$ for every multiplication gate

Key Requirements

- **Correctness:** $|\text{secret}| < P_{\text{all}}$
- **Efficiency:** $\log m = \Omega(\log P_{\text{all}})$

Overview of GJMW24: Preprocessing Phase

High Dimensional Smudging Lemma [GJMW24]

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$

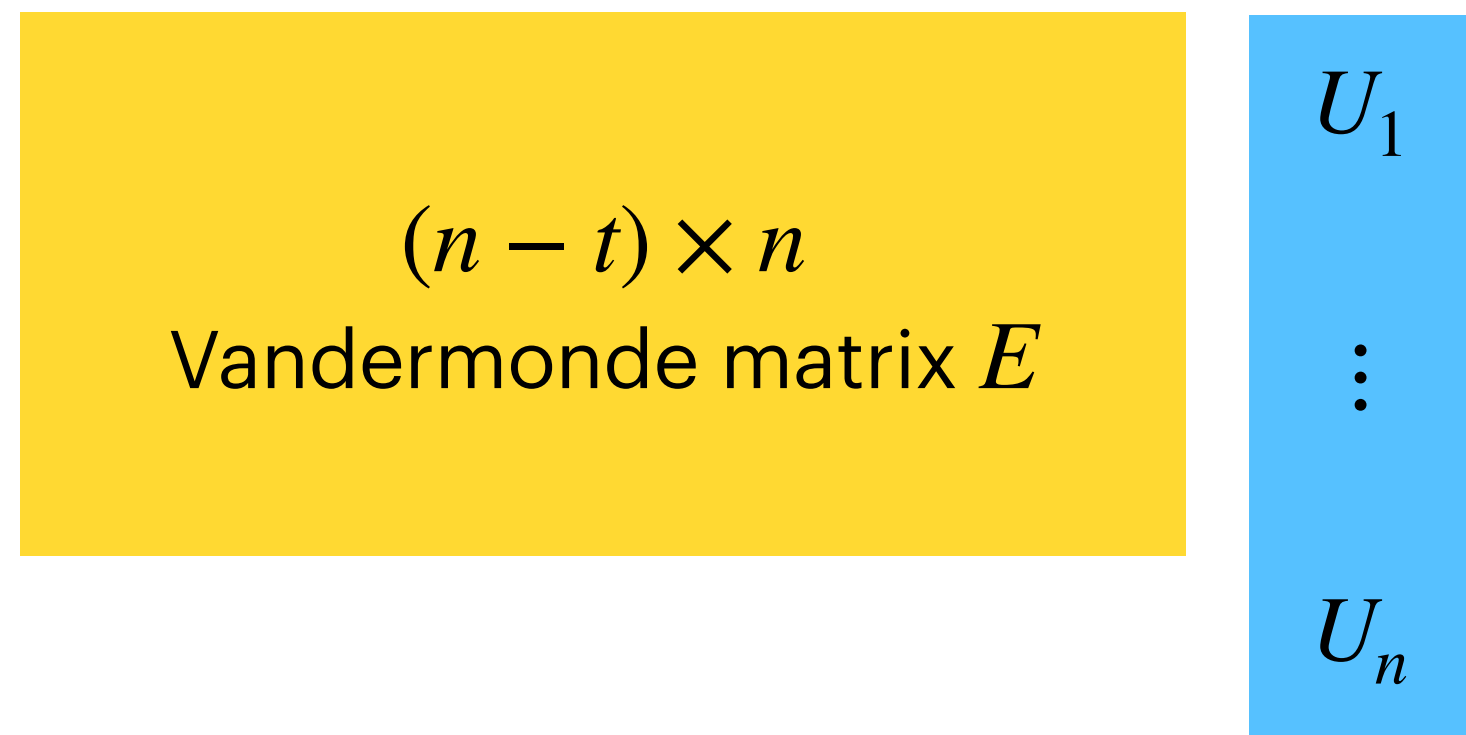
$$\begin{matrix} U_1 \\ \vdots \\ U_n \end{matrix}$$

where at least $n - t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$



where at least $n - t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$ and $Z_1, \dots, Z_{n-t} \leq m^2$

The diagram illustrates the equation: $(n-t) \times n$ Vandermonde matrix E plus a vector of values $U_1, \dots, U_n$ and $Z_1, \dots, Z_{n-t}$. The Vandermonde matrix E is represented by a yellow rectangle. The vector U is represented by a blue rectangle with U_1 at the top, U_n at the bottom, and a vertical ellipsis in the middle. The vector Z is represented by a red rectangle with Z_1 at the top, Z_{n-t} at the bottom, and a vertical ellipsis in the middle. A plus sign is placed between the blue and red rectangles.

$$\begin{matrix} (n-t) \times n \\ \text{Vandermonde matrix } E \end{matrix} + \begin{matrix} U_1 \\ \vdots \\ U_n \end{matrix} + \begin{matrix} Z_1 \\ \vdots \\ Z_{n-t} \end{matrix}$$

where at least $n-t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Overview of GJMW24: Preprocessing Phase

High Dimensional Smudging Lemma [GJMW24]

For sufficiently large $L \in \mathbb{N}$ and $Z_1, \dots, Z_{n-t} \leq m^2$

$$\begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array} + \begin{array}{c} Z_1 \\ \vdots \\ Z_{n-t} \end{array} \approx^s \begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array}$$

where at least $n - t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$ and $Z_1, \dots, Z_{n-t} \leq m^2$

$$\begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array} + \begin{array}{c} Z_1 \\ \vdots \\ Z_{n-t} \end{array} \approx^s \begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array}$$

where at least $n-t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Similar to linear randomness extraction in [Damgård-Nielsen'07]

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$ and $Z_1, \dots, Z_{n-t} \leq m^2$

$$\begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array} + \begin{array}{c} Z_1 \\ \vdots \\ Z_{n-t} \end{array} \approx^s \begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array}$$

where at least $n-t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Similar to linear randomness extraction in [Damgård-Nielsen'07]

Realizes preprocessing by batch extraction of integer masks

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$ and $Z_1, \dots, Z_{n-t} \leq m^2$

$(n - t) \times n$
Vandermonde matrix E

U_1
 $\vdots$
 U_n

$+$

Z_1
 $\vdots$
 Z_{n-t}

$\approx^s$

$(n - t) \times n$
Vandermonde matrix E

U_1
 $\vdots$
 U_n

where at least $n - t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Vandermonde matrix-vector product takes $O(n \log n)$ ops

Overview of GJM24: Preprocessing Phase

High Dimensional Smudging Lemma [GJM24]

For sufficiently large $L \in \mathbb{N}$ and $Z_1, \dots, Z_{n-t} \leq m^2$

$$\begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array} + \begin{array}{c} Z_1 \\ \vdots \\ Z_{n-t} \end{array} \approx^s \begin{array}{c} (n-t) \times n \\ \text{Vandermonde matrix } E \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array}$$

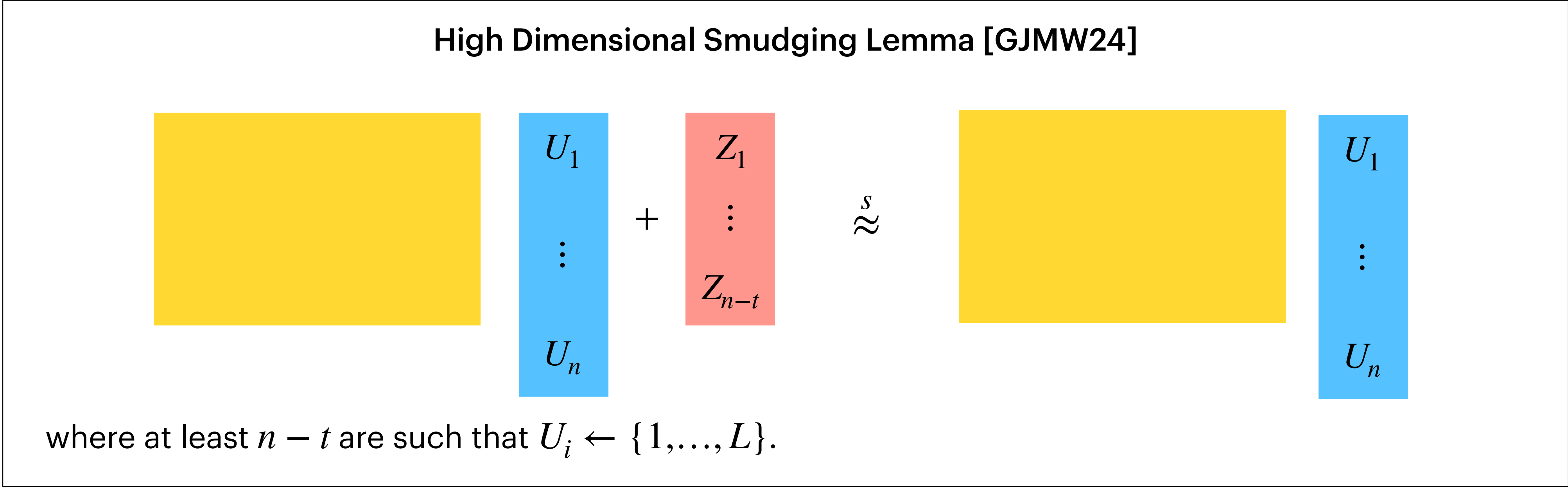
where at least $n-t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

Vandermonde matrix-vector product takes $O(n \log n)$ ops

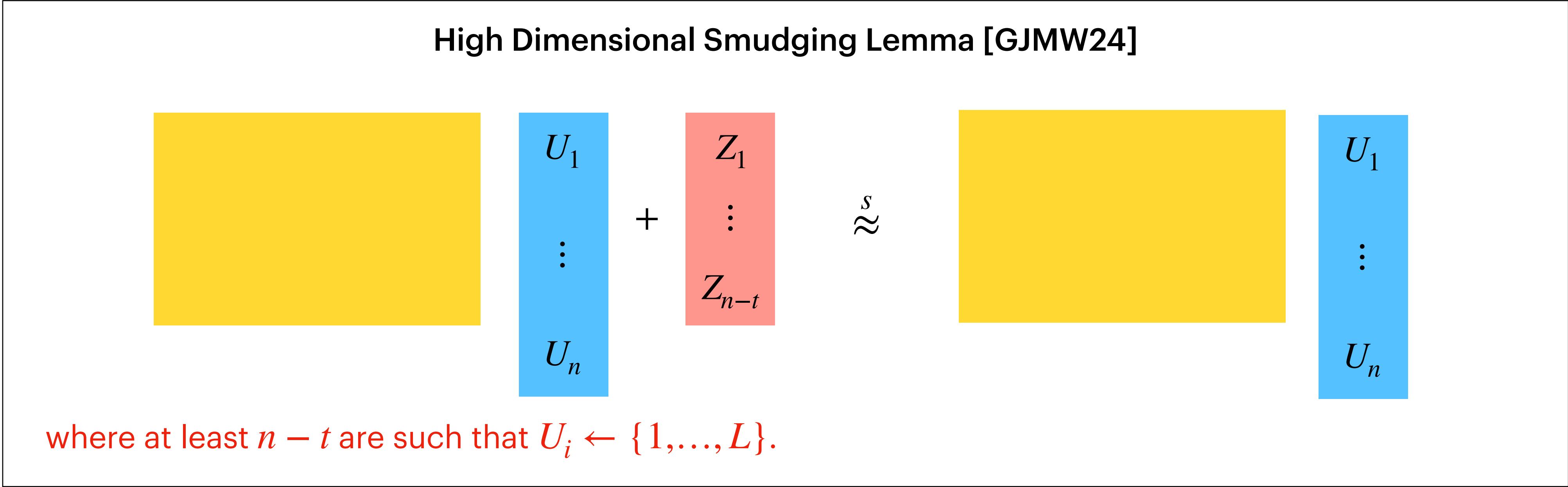
$O(\log m)$ amortized computation per preprocessing tuple

GJMW24 Requirement on Modulus Size

GJMW24 Requirement on Modulus Size



GJMW24 Requirement on Modulus Size



GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} \begin{array}{|c|c|} \hline E_H & E_I \\ \hline \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array} + \begin{array}{c} Z_1 \\ \vdots \\ Z_{n-t} \end{array} \approx^s \begin{array}{c} \begin{array}{|c|c|} \hline E_H & E_I \\ \hline \end{array} \begin{array}{c} U_1 \\ \vdots \\ U_n \end{array} \end{array}$$

where at least $n - t$ are such that $U_i \leftarrow \{1, \dots, L\}$.

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

E_H

U_1
 $\vdots$
 U_{n-t}

+

Z_1
 $\vdots$
 Z_{n-t}

$\approx^s$

E_H

U_1
 $\vdots$
 U_{n-t}

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

E_H

U_1
 $\vdots$
 U_{n-t}

+

Z_1
 $\vdots$
 Z_{n-t}

$\approx^s$

E_H

U_1
 $\vdots$
 U_{n-t}

Vandermonde matrix is **super-invertible**: every $(n - t) \times (n - t)$ submatrix is **non-singular**

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} E_H^{-1} \end{array} \begin{array}{c} Z_1 \\ \vdots \\ Z_{n-t} \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

Vandermonde matrix is **super-invertible**: every $(n - t) \times (n - t)$ submatrix is **non-singular**

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{pmatrix} U_1 \\ \vdots \\ U_{n-t} \end{pmatrix} + \begin{pmatrix} & \\ & E_H^{-1} \\ & \end{pmatrix} \begin{pmatrix} Z_1 \\ \vdots \\ Z_{n-t} \end{pmatrix} \approx^s \begin{pmatrix} U_1 \\ \vdots \\ U_{n-t} \end{pmatrix}$$

- While $\det(E_H) \neq 0$, E_H might not be invertible over the integers

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{pmatrix} U_1 \\ \vdots \\ U_{n-t} \end{pmatrix} + \begin{pmatrix} & \\ & E_H^{-1} \\ & \end{pmatrix} \begin{pmatrix} Z_1 \\ \vdots \\ Z_{n-t} \end{pmatrix} \approx^s \begin{pmatrix} U_1 \\ \vdots \\ U_{n-t} \end{pmatrix}$$

- While $\det(E_H) \neq 0$, E_H might not be invertible over the integers
- However, E_H is invertible over the rationals

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{pmatrix} U_1 \\ \vdots \\ U_{n-t} \end{pmatrix} + \begin{pmatrix} & \\ & E_H^{-1} \\ & \end{pmatrix} \begin{pmatrix} Z_1 \\ \vdots \\ Z_{n-t} \end{pmatrix} \approx^s \begin{pmatrix} U_1 \\ \vdots \\ U_{n-t} \end{pmatrix}$$

- While $\det(E_H) \neq 0$, E_H might not be invertible over the integers
- However, E_H is invertible over the rationals

$$E_H^{-1} = \frac{\text{adj}(E_H)}{\det(E_H)}$$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- While $\det(E_H) \neq 0$, E_H might not be invertible over the integers
- However, E_H is invertible over the rationals

$$E_H^{-1} = \frac{\text{adj}(E_H)}{\det(E_H)}$$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

U_1
 $\vdots$
 U_{n-t}

+

$\text{adj}(E_H)$

$Z_1 / \det(E_H)$
 $\vdots$
 $Z_{n-t} / \det(E_H)$

$\approx^s$

U_1
 $\vdots$
 U_{n-t}

- Therefore, each secret Z_i needs to be divisible by $\det(E_H)$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Therefore, each secret Z_i needs to be divisible by $\det(E_H)$
 $\Lambda = \text{LCM of } \det(E_H) \text{ over all } H, |H| = n - t$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Therefore, each secret Z_i needs to be divisible by $\det(E_H)$
 $\Lambda = \text{LCM of } \det(E_H) \text{ over all } H, |H| = n - t$
- GJMW24 **reformats secrets**: given z compute Z such that

$$z = Z \bmod m \quad \text{and} \quad Z \equiv 0 \pmod{\Lambda}$$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \stackrel{s}{\approx} \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Therefore, each secret Z_i needs to be divisible by $\det(E_H)$
 $\Lambda = \text{LCM of } \det(E_H) \text{ over all } H, |H| = n - t$
- GJMW24 **reformats secrets**: given z compute Z such that

$$z = Z \bmod m \quad \text{and} \quad Z \equiv 0 \pmod{\Lambda}$$

- For Vandermonde matrices, $\Lambda = n^{\Omega(n^2)}$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Therefore, each secret Z_i needs to be divisible by $\det(E_H)$
 $\Lambda = \text{LCM of } \det(E_H) \text{ over all } H, |H| = n - t$
- GJMW24 **reformats secrets**: given z compute Z such that

$$z = Z \bmod m \quad \text{and} \quad Z \equiv 0 \pmod{\Lambda}$$

- For Vandermonde matrices, $\Lambda = n^{\Omega(n^2)}$
- Secrets are as large as $n^{\Omega(n^2)}$

Key Requirements

- **Correctness:** $|\text{secret}| < P_{\text{all}}$
- **Efficiency:** $\log m = \Omega(\log P_{\text{all}})$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Therefore, each secret Z_i needs to be divisible by $\det(E_H)$
 $\Lambda = \text{LCM of } \det(E_H) \text{ over all } H, |H| = n - t$
- GJMW24 **reformats secrets**: given z compute Z such that

$$z = Z \bmod m \quad \text{and} \quad Z \equiv 0 \pmod{\Lambda}$$

- For Vandermonde matrices, $\Lambda = n^{\Omega(n^2)}$
- Secrets are as large as $n^{\Omega(n^2)}$

Key Requirements

- **Correctness:** $|\text{secret}| < P_{\text{all}}$
- **Efficiency:** $\log m = \Omega(\log P_{\text{all}})$

$$\text{Therefore, } \log m = \Omega(n^2 \log n)$$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

U_1
 $\vdots$
 U_{n-t}

+

$\text{adj}(E_H)$

$Z_1 / \det(E_H)$
 $\vdots$
 $Z_{n-t} / \det(E_H)$

$\approx^s$

U_1
 $\vdots$
 U_{n-t}

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

U_1
 $\vdots$
 U_{n-t}

+

$\text{adj}(E_H)$

$Z_1 / \det(E_H)$
 $\vdots$
 $Z_{n-t} / \det(E_H)$

$\approx^s$

U_1
 $\vdots$
 U_{n-t}

- Statistical closeness requires

$$U_i \gg \|\text{adj}(E_H)\|_\infty \approx |\det(E_H)|$$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \stackrel{s}{\approx} \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Statistical closeness requires

$$U_i \gg \|\text{adj}(E_H)\|_\infty \approx |\det(E_H)|$$

- For Vandermonde matrices, $|\det(E_H)| = n^{\Omega(n^2)}$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Statistical closeness requires

$$U_i \gg \|\text{adj}(E_H)\|_\infty \approx |\det(E_H)|$$

- For Vandermonde matrices, $|\det(E_H)| = n^{\Omega(n^2)}$
- Therefore, $U_i \gg n^{\Omega(n^2)}$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Statistical closeness requires

$$U_i \gg \|\text{adj}(E_H)\|_\infty \approx |\det(E_H)|$$

- For Vandermonde matrices, $|\det(E_H)| = n^{\Omega(n^2)}$
- Therefore, $U_i \gg n^{\Omega(n^2)}$

Key Requirements

- Correctness:** $|\text{secret}| < P_{\text{all}}$
- Efficiency:** $\log m = \Omega(\log P_{\text{all}})$

GJMW24 Requirement on Modulus Size

High Dimensional Smudging Lemma [GJMW24]

$$\begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array} + \begin{array}{c} \text{adj}(E_H) \end{array} \begin{array}{c} Z_1 / \det(E_H) \\ \vdots \\ Z_{n-t} / \det(E_H) \end{array} \approx^s \begin{array}{c} U_1 \\ \vdots \\ U_{n-t} \end{array}$$

- Statistical closeness requires

$$U_i \gg \|\text{adj}(E_H)\|_\infty \approx |\det(E_H)|$$

- For Vandermonde matrices, $|\det(E_H)| = n^{\Omega(n^2)}$
- Therefore, $U_i \gg n^{\Omega(n^2)}$

Key Requirements

- Correctness:** $|\text{secret}| < P_{\text{all}}$
- Efficiency:** $\log m = \Omega(\log P_{\text{all}})$

$$\text{Therefore, } \log m = \Omega(n^2 \log n)$$

Our Approach

Why does GJMW24 require $\log m = \Omega(n^2 \log n)$?

1. **Secret reformatting:** Proof requires secrets to be a multiple of “LCM of determinants”
2. **Large determinants:** Need to mask secrets as large as the determinant which is $n^{\Omega(n^2)}$

Our Approach

Why does GJMW24 require $\log m = \Omega(n^2 \log n)$?

1. **Secret reformatting:** Proof requires secrets to be a multiple of “LCM of determinants”
2. **Large determinants:** Need to mask secrets as large as the determinant which is $n^{\Omega(n^2)}$

Our Contribution: General framework for linear “randomness” extractors over the integers

Our Approach

Why does GJM24 require $\log m = \Omega(n^2 \log n)$?

1. **Secret reformatting:** Proof requires secrets to be a multiple of “LCM of determinants”
2. **Large determinants:** Need to mask secrets as large as the determinant which is $n^{\Omega(n^2)}$

New proof strategy that avoids secret reformatting

Our Contribution: General framework for linear “randomness” extractors over the integers

Our Approach

Why does GJM24 require $\log m = \Omega(n^2 \log n)$?

1. **Secret reformatting:** Proof requires secrets to be a multiple of “LCM of determinants”
2. **Large determinants:** Need to mask secrets as large as the determinant which is $n^{\Omega(n^2)}$

New proof strategy that avoids secret reformatting

Matrices with determinants bounded by $n^{O(n)}$

Our Contribution: General framework for linear “randomness” extractors over the integers

Our Approach

Why does GJM24 require $\log m = \Omega(n^2 \log n)$?

1. **Secret reformatting:** Proof requires secrets to be a multiple of “LCM of determinants”

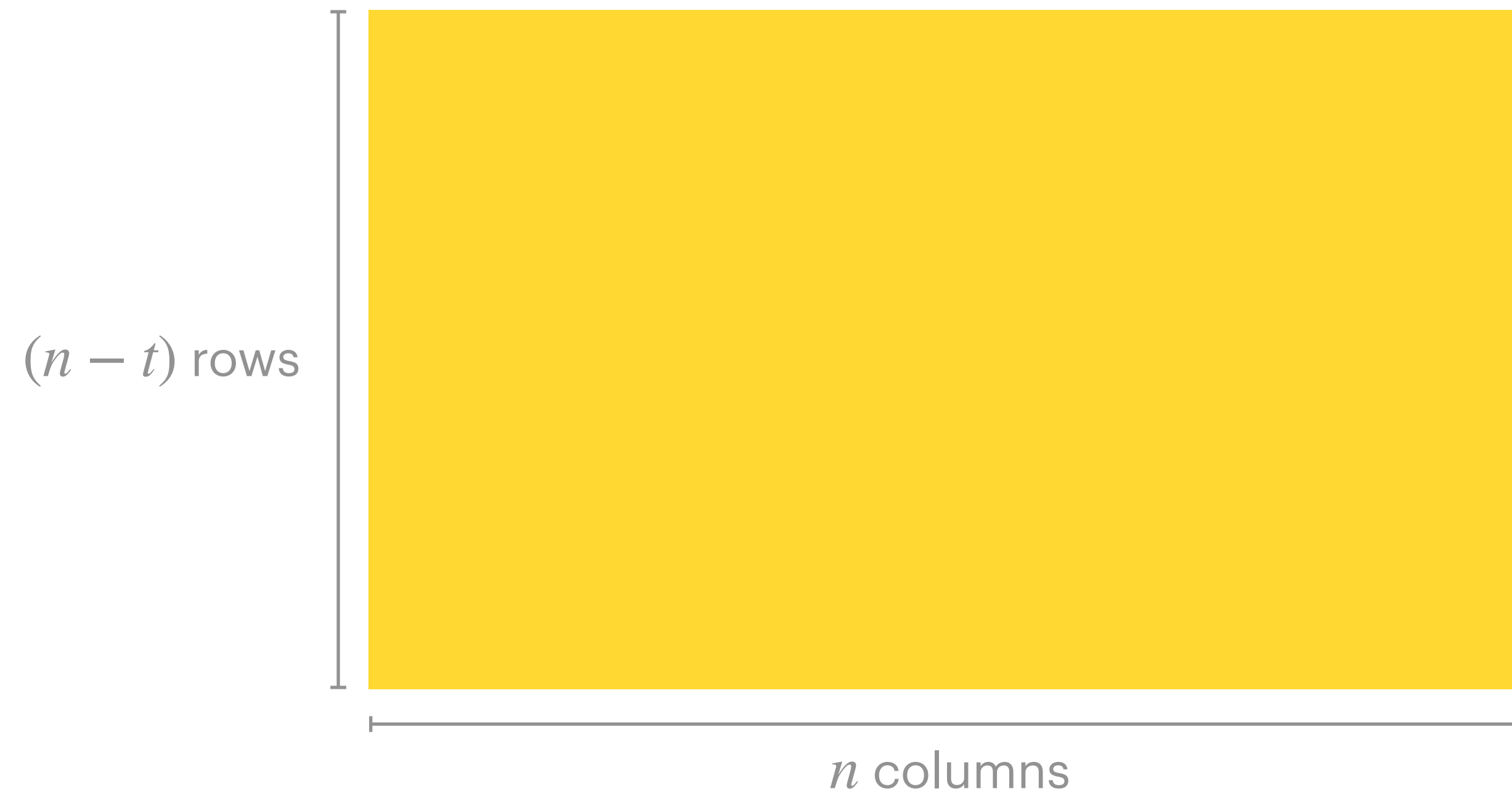
2. **Large determinants:** Need to mask secrets as large as the determinant which is $n^{\Omega(n^2)}$

New proof strategy that avoids secret reformatting

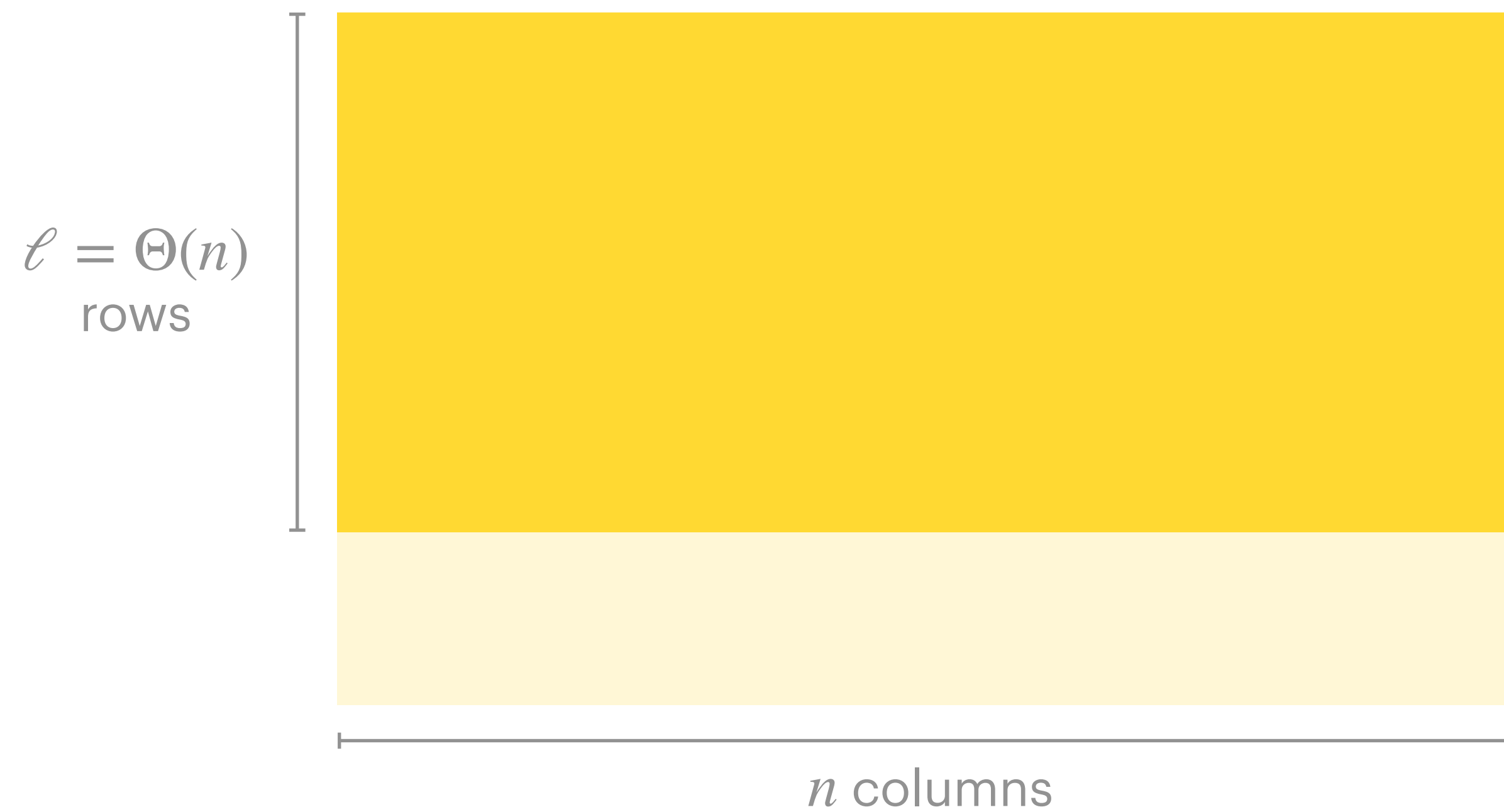
Matrices with determinants bounded by $n^{O(n)}$

Our Contribution: General framework for linear “randomness” extractors over the integers

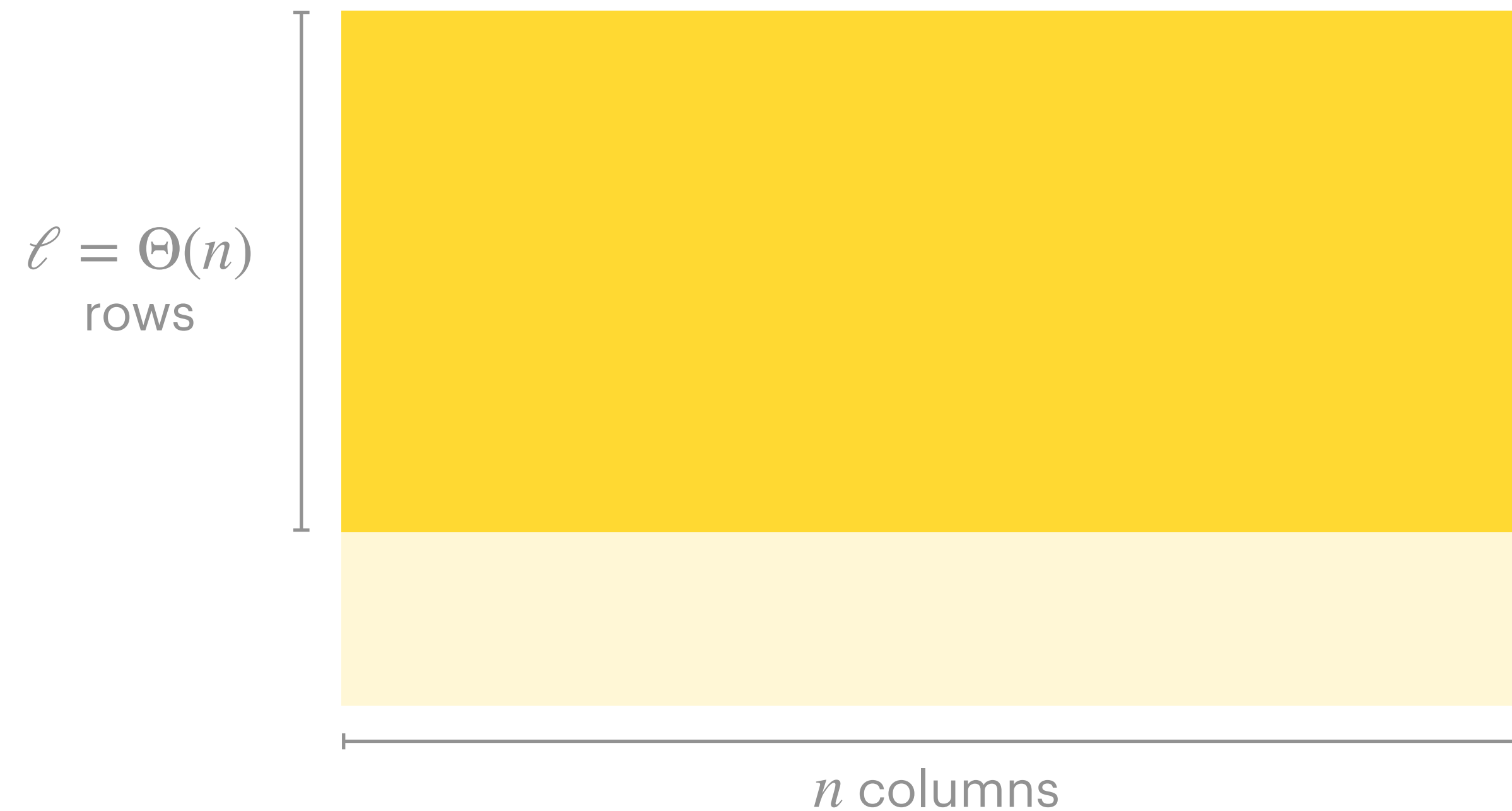
Our Approach: Avoiding Secret Reformatting



Our Approach: Avoiding Secret Reformatting



Our Approach: Avoiding Secret Reformatting

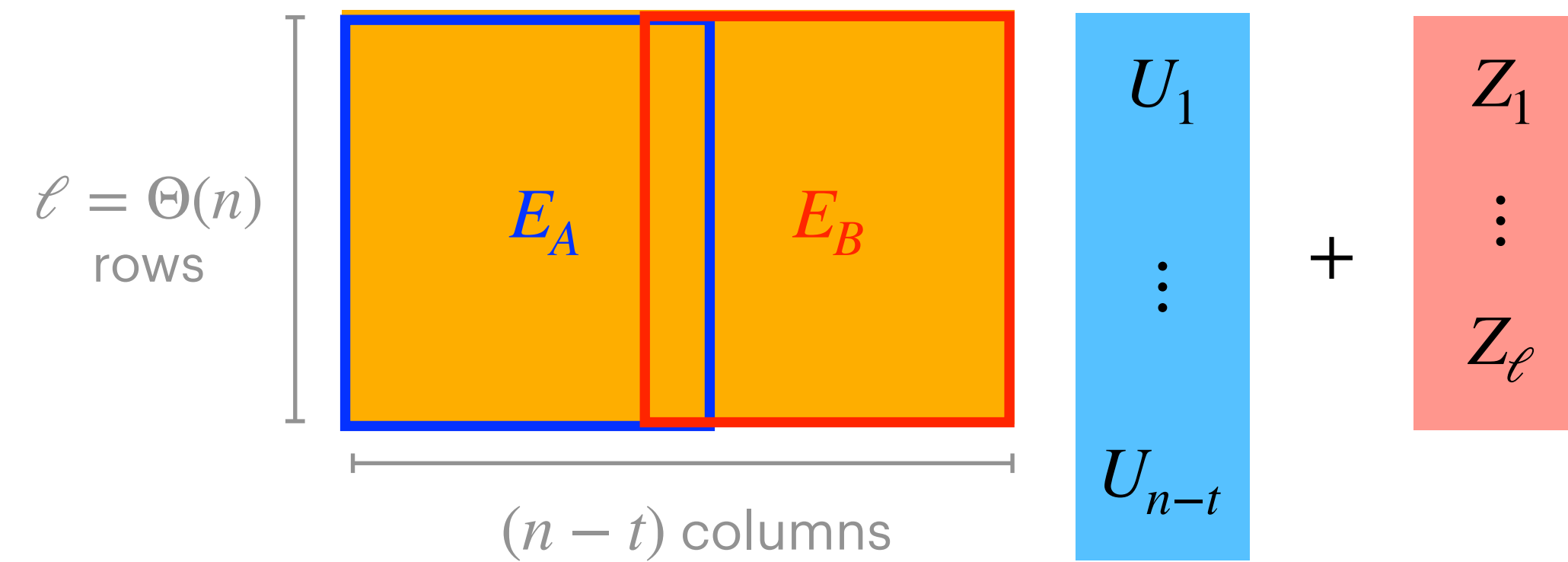


Secret reformatting can be avoided if there are maximal submatrices with co-prime determinants

Our Approach: Avoiding Secret Reformatting



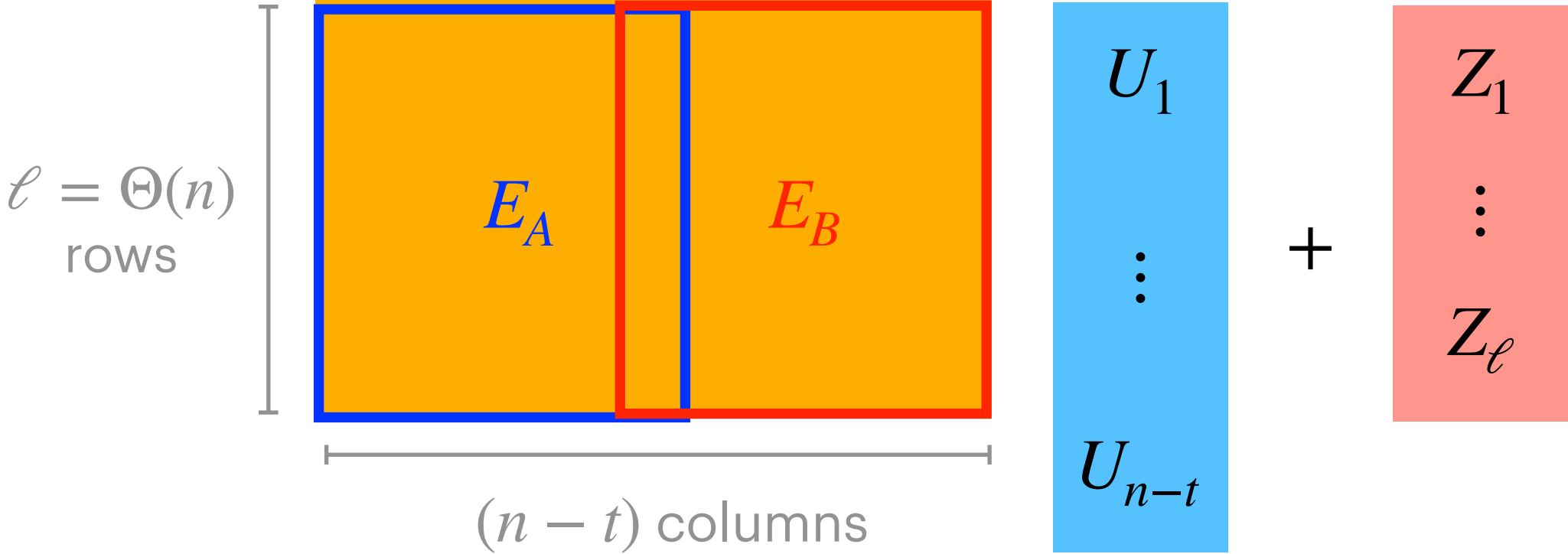
Our Approach: Avoiding Secret Reformatting



Let E_A and E_B be $\ell \times \ell$ matrices such that

$$\gcd \left(\det(E_A), \det(E_B) \right) = 1$$

Our Approach: Avoiding Secret Reformatting



Let E_A and E_B be $\ell \times \ell$ matrices such that

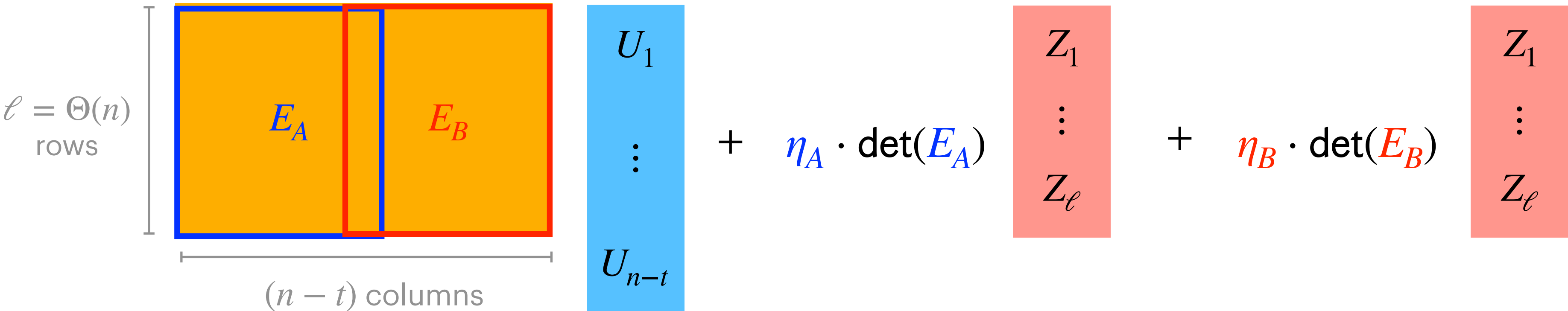
$$\gcd \left(\det(E_A), \det(E_B) \right) = 1$$



Exists Bézout coefficients η_A and η_B such that

$$\eta_A \cdot \det(E_A) + \eta_B \cdot \det(E_B) = 1$$

Our Approach: Avoiding Secret Reformatting



Let E_A and E_B be $\ell \times \ell$ matrices such that

$$\gcd \left(\det(E_A), \det(E_B) \right) = 1$$



Exists Bézout coefficients η_A and η_B such that

$$\eta_A \cdot \det(E_A) + \eta_B \cdot \det(E_B) = 1$$

Our Approach: Avoiding Secret Reformatting

The diagram illustrates a matrix equation used to avoid secret reformatting. It consists of the following components:

- Matrix Dimensions:** The matrix has $\ell = \Theta(n)$ rows and $(n - t)$ columns.
- Matrix Structure:** The matrix is composed of two blocks: E_A (orange square) and E_B (yellow rectangle).
- Vector U :** A vertical vector U is shown to the right of the matrix, with the top part labeled U_1 and the bottom part labeled U_{n-t} .
- Equation:** The matrix equation is:
$$\begin{bmatrix} E_A & E_B \end{bmatrix} \begin{bmatrix} U_1 \\ \vdots \\ U_{n-t} \end{bmatrix} + \eta_A \cdot \det(E_A) \begin{bmatrix} Z_1 \\ \vdots \\ Z_\ell \end{bmatrix} + \eta_B \cdot \det(E_B) \begin{bmatrix} Z_1 \\ \vdots \\ Z_\ell \end{bmatrix}$$

Our Approach: Avoiding Secret Reformatting

The diagram illustrates a matrix decomposition. On the left, a large matrix is shown with a blue border. It is divided into two main horizontal sections. The top section is a square block labeled E_A in blue, with a vertical dimension label $\ell = \Theta(n)$ rows to its left. To the right of E_A is a rectangular block labeled E_B in pink. Below these two blocks, a horizontal dimension label $(n - t)$ columns is shown. To the right of the E_B block is a vertical blue column labeled U_1 at the top and U_{n-t} at the bottom. This is followed by a plus sign and a blue scalar term $\eta_A \cdot \det(E_A)$. To the right of this term is a vertical red column labeled Z_1 at the top, followed by a vertical ellipsis, and then Z_ℓ at the bottom. This is followed by another plus sign and a pink scalar term $\eta_B \cdot \det(E_B)$. To the right of this term is a vertical pink column labeled Z_1 at the top, followed by a vertical ellipsis, and then Z_ℓ at the bottom.

$$\begin{matrix} \ell = \Theta(n) \text{ rows} \\ \begin{bmatrix} E_A & E_B \end{bmatrix} \begin{matrix} U_1 \\ \vdots \\ U_{n-t} \end{matrix} + \eta_A \cdot \det(E_A) \begin{matrix} Z_1 \\ \vdots \\ Z_\ell \end{matrix} + \eta_B \cdot \det(E_B) \begin{matrix} Z_1 \\ \vdots \\ Z_\ell \end{matrix} \end{matrix}$$

$(n - t)$ columns

We always mask a multiple of the determinant

Our Approach: Avoiding Secret Reformatting

The diagram illustrates a matrix decomposition. On the left, a large matrix is shown with dimensions $\ell = \Theta(n)$ rows and $(n - t)$ columns. This matrix is composed of two parts: a blue square block labeled E_A and a yellow rectangular block labeled E_B . To the right of this matrix is a blue vertical column vector labeled U_1 at the top and U_{n-t} at the bottom. This is followed by a plus sign and a blue scalar term $\eta_A \cdot \det(E_A)$. Next is a red vertical column vector labeled Z_1 at the top, followed by a vertical ellipsis, and Z_ℓ at the bottom. This is followed by another plus sign and a red scalar term $\eta_B \cdot \det(E_B)$. Finally, there is a light red vertical column vector labeled Z_1 at the top, followed by a vertical ellipsis, and Z_ℓ at the bottom.

We always mask a multiple of the determinant

Can use an argument similar to GJM24 [without reformatting](#)

Our Approach: Avoiding Secret Reformatting

The diagram illustrates a matrix equation. On the left, a matrix is shown with two blocks: an orange square block labeled E_A and a yellow rectangular block labeled E_B . A vertical dimension line to the left of E_A is labeled $\ell = \Theta(n)$ rows. A horizontal dimension line below both E_A and E_B is labeled $(n - t)$ columns. To the right of E_B is a blue vertical column vector labeled U_1 at the top and U_{n-t} at the bottom, with vertical ellipsis dots in between. This is followed by a plus sign and a blue scalar term $\eta_A \cdot \det(E_A)$. Next is a red vertical column vector labeled Z_1 at the top and Z_ℓ at the bottom, with vertical ellipsis dots in between. This is followed by a plus sign and a red scalar term $\eta_B \cdot \det(E_B)$. Finally, there is a light red vertical column vector labeled Z_1 at the top and Z_ℓ at the bottom, with vertical ellipsis dots in between.

We always mask a multiple of the determinant

Can use an argument similar to GJM24 [without reformatting](#)

Bézout coefficients need to be bounded to ensure small U_i

Our Approach: Ensuring Efficiency

Matrix needs to support fast matrix-vector product

Our Approach: Ensuring Efficiency

Matrix needs to support fast matrix-vector product



Toeplitz matrix where each diagonal value is sampled uniformly from $\{1, \dots, n\}$

Our Approach: Ensuring Efficiency

Matrix needs to support fast matrix-vector product



Toeplitz matrix where each diagonal value is sampled uniformly from $\{1, \dots, n\}$

Random Toeplitz matrices satisfy all required properties

Our Approach: Ensuring Efficiency

Matrix needs to support fast matrix-vector product



Toeplitz matrix where each diagonal value is sampled uniformly from $\{1, \dots, n\}$

Random Toeplitz matrices satisfy all required properties

Determinants of maximal square submatrices are bounded by $n^{O(n)}$

Our Approach: Ensuring Efficiency

Matrix needs to support fast matrix-vector product



Toeplitz matrix where each diagonal value is sampled uniformly from $\{1, \dots, n\}$

Random Toeplitz matrices satisfy all required properties

Determinants of maximal square submatrices are **bounded by $n^{O(n)}$**

Submatrix with $(n - t)$ columns has square submatrices with **co-prime determinants**

Our Approach: Ensuring Efficiency

Matrix needs to support fast matrix-vector product



Toeplitz matrix where each diagonal value is sampled uniformly from $\{1, \dots, n\}$

Random Toeplitz matrices satisfy all required properties

Determinants of maximal square submatrices are **bounded by $n^{O(n)}$**

Submatrix with $(n - t)$ columns has square submatrices with **co-prime determinants**

Toeplitz matrices admit **$O(n \log m)$ time matrix-vector product** [Feist–Khovratovich'23]

Conclusion

Conclusion

- Proving Toeplitz matrices consist of submatrices with co-prime determinants after removing any t columns requires carefully applying the union bound over all choices of t columns.

Conclusion

- Proving Toeplitz matrices consist of submatrices with co-prime determinants after removing any t columns requires carefully applying the union bound over all choices of t columns.
- Open Questions

Conclusion

- Proving Toeplitz matrices consist of submatrices with co-prime determinants after removing any t columns requires carefully applying the union bound over all choices of t columns.
- **Open Questions**
 - GJM24 provides MPC with minimal overhead for dishonest majority setting assuming preprocessing. Can this preprocessing be realized with minimal overhead?

Conclusion

- Proving Toeplitz matrices consist of submatrices with **co-prime determinants** after **removing any t columns** requires carefully applying the **union bound over all choices of t columns**.
- **Open Questions**
 - GJM24 provides MPC with minimal overhead for **dishonest majority** setting **assuming preprocessing**. Can this preprocessing be realized with minimal overhead?
 - Other techniques for MPC with minimal overhead?

Conclusion

- Proving Toeplitz matrices consist of submatrices with **co-prime determinants** after **removing any t columns** requires carefully applying the **union bound over all choices of t columns**.
- **Open Questions**
 - GJM24 provides MPC with minimal overhead for **dishonest majority** setting **assuming preprocessing**. Can this preprocessing be realized with minimal overhead?
 - Other techniques for MPC with minimal overhead?
 - **Existing Work:**
 - Amortizing over parallelization in the circuit (SIMD)
 - Amortizing over size of the ring (this work)

Conclusion

- Proving Toeplitz matrices consist of submatrices with **co-prime determinants** after **removing any t columns** requires carefully applying the **union bound over all choices of t columns**.
- **Open Questions**
 - GJM24 provides MPC with minimal overhead for **dishonest majority** setting **assuming preprocessing**. Can this preprocessing be realized with minimal overhead?
 - Other techniques for MPC with minimal overhead?
 - **Existing Work:**
 - Amortizing over parallelization in the circuit (SIMD)
 - Amortizing over size of the ring (this work)

Thank You