Faster Sounder Proofs

Justin Holmgren
NTT Research

Ron Rothblum
Technion

Paper

Slides

Faster (Sounder Proofs)

Justin Holmgren
NTT Research

Ron Rothblum
Technion

Paper

Slides

Proof Systems

Proof Systems: the Theory Perspective

Proof Systems: the Theory Perspective

Super-efficient verification of complex statements:

- Evolution: NP proof, interactive proofs, MIP, PCP, …
- Central objects in theory literature: NP completeness, zero-knowledge, PCP Theorem, …

Proof Systems: the Theory Perspective

Super-efficient verification of complex statements:

- Evolution: NP proof, interactive proofs, MIP, PCP, …
- Central objects in theory literature: NP completeness, zero-knowledge, PCP Theorem, …

Proofs spurred the development of lots of deep theory!

Practical Motivation:
Blockchains & Cryptocurrencies

Efficient ZK proofs have amazing applications.

"Perhaps the most powerful cryptographic technology to come out of the last decade is general-purpose succinct zero knowledge proofs"

- Vitalik Buterin, Ethereum co-founder (2021)

Complexity of Proving

Key efficiency bottleneck: complexity of *generating the proof*.

In practice: orders of magnitude slower than just performing the computation.

Complexity of Proving

Key efficiency bottleneck: complexity of *generating the proof*.

In practice: orders of magnitude slower than just performing the computation.

Can proving be as fast as computing?

Complexity of Proving

Key efficiency bottleneck: complexity of *generating the proof*.

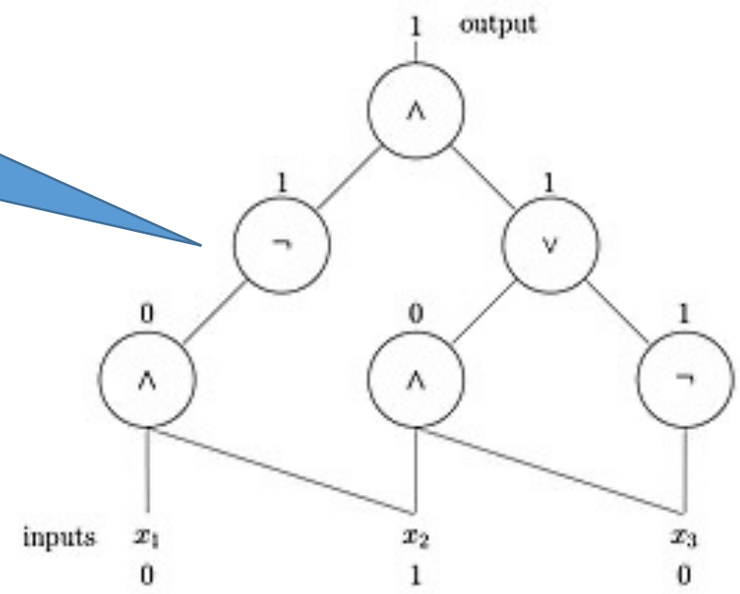
In practice: orders of magnitude slower than just performing the computation.

Can proving be as fast as computing?

Looking at very small factors ⇒ the computational model matters!

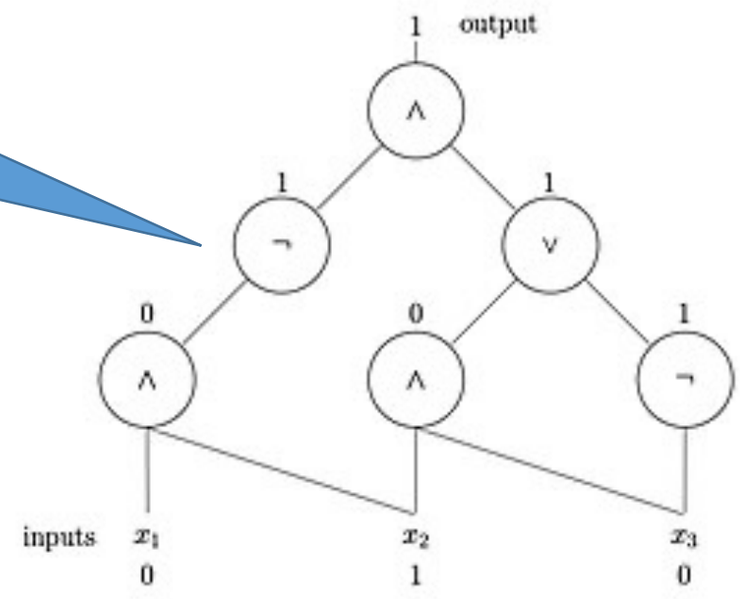
Today: Boolean Circuits

Constant fan-in, arbitrary gates



Today: Boolean Circuits

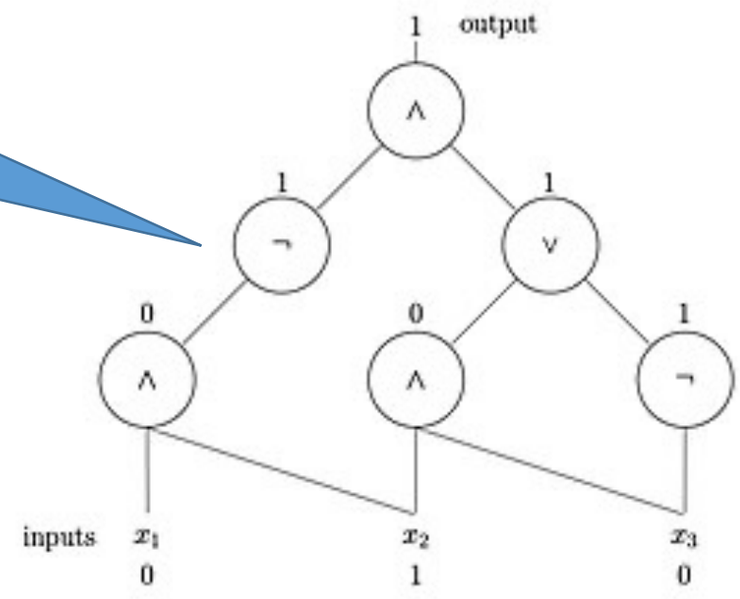
Constant fan-in, arbitrary gates



Why Boolean circuits? [IKOS08]

Today: Boolean Circuits

Constant fan-in, arbitrary gates



Why Boolean circuits? [IKOS08]

- Natural, fundamental, and physically motivated model of computational complexity
- Includes interesting non-arithmetic computation (e.g, SHA).

Prover Complexity for Succinct Proofs:

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Prover Complexity for Succinct Proofs:

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Type of circuit K	Prover Complexity
Boolean	Boolean size $O(S \cdot \lambda)$ [RR21]

Prover Complexity for Succinct Proofs:

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Type of circuit K	Prover Complexity
Boolean	Boolean size $O(S \cdot \lambda)$ [RR21]

λ comes from parallel repetition

Prover Complexity for Succinct Proofs:

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Type of circuit K	Prover Complexity
Boolean	Boolean size $O(S \cdot \lambda)$ [RR21]
arithmetic over **large** field $\mathbb{F}$: $\lvert\mathbb{F}\rvert \geq S \cdot 2^{\lambda}$	arithmetic size $O(S)$ over $\mathbb{F}$ [BCGGHJ17,XZZPS19,ZWZZ20, BCG20,BCL20,GLSTW21]

λ comes from parallel repetition

Main Result

TL;DR: succinct proofs for "nice circuit" satisfiability with $S \cdot \text{polylog}(\lambda)$ size prover, soundness error $2^{-\lambda}$, and sublinear verification

hiding additive $\text{poly}(\lambda)$

exact complexity depends on niceness

Nice Boolean Circuits

Nice Boolean Circuits

- **Simple examples**: C is nice if:

Nice Boolean Circuits

- **Simple examples**: C is nice if:
  - it has batch structure:
    $C(x_1, \ldots, x_k) = D(x_1) \wedge \cdots \wedge D(x_k)$ for large k and some repeated sub-circuit D;

Nice Boolean Circuits

- **Simple examples**: C is nice if:
  - it has batch structure:
    $C(x_1, \ldots, x_k) = D(x_1) \wedge \cdots \wedge D(x_k)$ for large k and some repeated sub-circuit D;
  - or it has iterated structure:
    $C(x) = D(D \cdots (D(x) \cdots))$.

Nice Boolean Circuits

- **Simple examples**: C is nice if:
  - it has batch structure:
    $C(x_1, \ldots, x_k) = D(x_1) \wedge \cdots \wedge D(x_k)$ for large k and some repeated sub-circuit D;
  - or it has iterated structure:
    $C(x) = D(D \cdots (D(x) \cdots))$.
  - Our basic verifier has size $\approx |D|$; can improve with composition

Nice Boolean Circuits

- **Simple examples**: C is nice if:
  - it has batch structure:
    $C(x_1, \ldots, x_k) = D(x_1) \wedge \cdots \wedge D(x_k)$ for large k and some repeated sub-circuit D;
  - or it has iterated structure:
    $C(x) = D(D \cdots (D(x) \cdots))$.
  - Our basic verifier has size $\approx |D|$; can improve with composition
- **More generally,** we define a "tensor CSP", and say that C is nice if it has a small tensor CSP. Ask me for details, if you dare :D

Main Result

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Main Result

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Type of circuit K	Prover Complexity
Boolean	Boolean size $O(S \cdot \lambda)$ [RR21]

Main Result

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Type of circuit K	Prover Complexity
Boolean	Boolean size $O(S \cdot \lambda)$ [RR21]
arithmetic over **large** field $\mathbb{F}$: $\lvert\mathbb{F}\rvert \geq S \cdot 2^{\lambda}$	arithmetic size $O(S)$ over $\mathbb{F}$ [BCGGHJ17,XZZPS19,ZWZZ20, BCG20,BCL20,GLSTW21]

Main Result

Parameterized by:

circuit size S and soundness error $2^{-\lambda}$ (usually $\log S < \lambda \ll S$)

Type of circuit K	Prover Complexity
Boolean	Boolean size $O(S \cdot \lambda)$ [RR21]
"nice" Boolean (repetitive structure)	Boolean size $S \cdot \text{polylog}(\lambda)$ [this work]
arithmetic over **large** field $\mathbb{F}$: $\lvert\mathbb{F}\rvert \geq S \cdot 2^{\lambda}$	arithmetic size $O(S)$ over $\mathbb{F}$ [BCGGHJ17,XZZPS19,ZWZZ20, BCG20,BCL20,GLSTW21]

Core Lemma
(Encoded Multiplication IOP)

Core Lemma
(Encoded Multiplication IOP)

There exists a **linear code** $E : \mathbb{F}_2^N \to \mathbb{F}_2^{O(N)}$ such that:

Core Lemma (Encoded Multiplication IOP)

There exists a **linear code** $E : \mathbb{F}_2^N \to \mathbb{F}_2^{O(N)}$ such that:

- E is encodable by size $N \cdot \text{polylog}(\lambda)$ Boolean circuit.

Core Lemma (Encoded Multiplication IOP)

There exists a **linear code** $E : \mathbb{F}_2^N \to \mathbb{F}_2^{O(N)}$ such that:

- E is encodable by size $N \cdot \text{polylog}(\lambda)$ Boolean circuit.

- Given $E(X)$, $E(Y)$, and $E(Z)$, an **IOP prover** of size $N \cdot \text{polylog}(\lambda)$ can prove $Z = X \odot Y$ with soundness error $2^{-\lambda}$.

Core Lemma (Encoded Multiplication IOP)

There exists a **linear code** $E : \mathbb{F}_2^N \to \mathbb{F}_2^{O(N)}$ such that:

- E is encodable by size $N \cdot \text{polylog}(\lambda)$ Boolean circuit.

Enforceable promise

- Given $E(X)$, $E(Y)$, and $E(Z)$, an **IOP prover** of size $N \cdot \text{polylog}(\lambda)$ can prove $Z = X \odot Y$ with soundness error $2^{-\lambda}$.

Core Lemma (Encoded Multiplication IOP)

There exists a **linear code** $E : \mathbb{F}_2^N \to \mathbb{F}_2^{O(N)}$ such that:

- E is encodable by size $N \cdot \text{polylog}(\lambda)$ Boolean circuit.

Enforceable promise

- Given $E(X)$, $E(Y)$, and $E(Z)$, an **IOP prover** of size $N \cdot \text{polylog}(\lambda)$ can prove $Z = X \odot Y$ with soundness error $2^{-\lambda}$.

Pointwise multiplication

Bonus Time:

Technical Tour

Off-The-Shelf Ingredients

Off-The-Shelf Ingredients

- Field $\mathbb{F}$ with $|\mathbb{F}| = O(\lambda)$

Off-The-Shelf Ingredients

- Field $\mathbb{F}$ with $|\mathbb{F}| = O(\lambda)$
- Good, linear, systematic codes.

Off-The-Shelf Ingredients

- Field $\mathbb{F}$ with $|\mathbb{F}| = O(\lambda)$
- Good, linear, systematic codes.
- $G : \mathbb{F}^n \to \mathbb{F}^{O(n)}$ encodable in size $O(n)$ such that:

  Given $G(x)$, $G(y)$, and $G(z)$, a prover can convince that $z = x \odot y$ with $O(1)$ soundness error and $O(1)$ prover overhead [Ron-Zewi Rothblum (STOC '22)]

Off-The-Shelf Ingredients

- Field $\mathbb{F}$ with $|\mathbb{F}| = O(\lambda)$
- Good, linear, systematic codes.
- $G : \mathbb{F}^n \to \mathbb{F}^{O(n)}$ encodable in size $O(n)$ such that:

  Given $G(x)$, $G(y)$, and $G(z)$, a prover can convince that $z = x \odot y$ with $O(1)$ soundness error and $O(1)$ prover overhead [Ron-Zewi Rothblum (STOC '22)]

- $M : \mathbb{F}^\lambda \to \mathbb{F}^{O(\lambda)}$ is a **multiplication code** encodable in size $\lambda \cdot \text{polylog}(\lambda)$ (e.g. Reed-Solomon + FFT)

Final Boss Slide

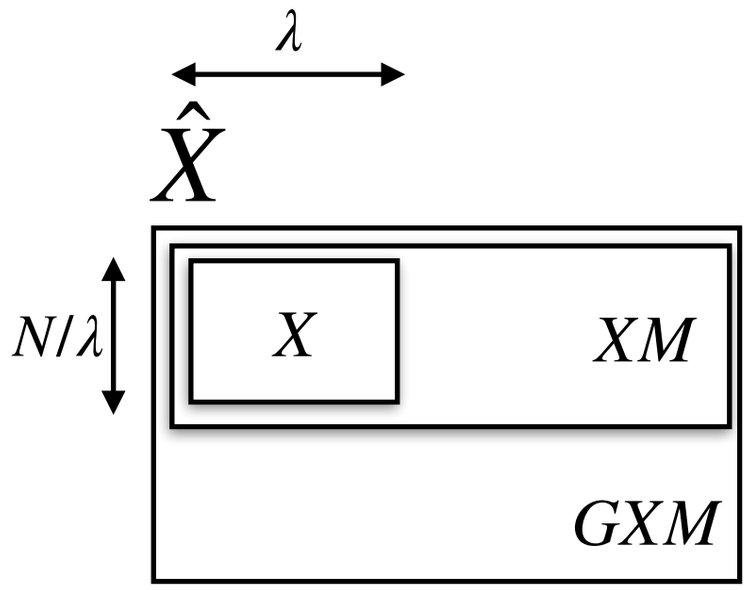
Pointwise Multiplication IOP

Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

Pointwise Multiplication IOP

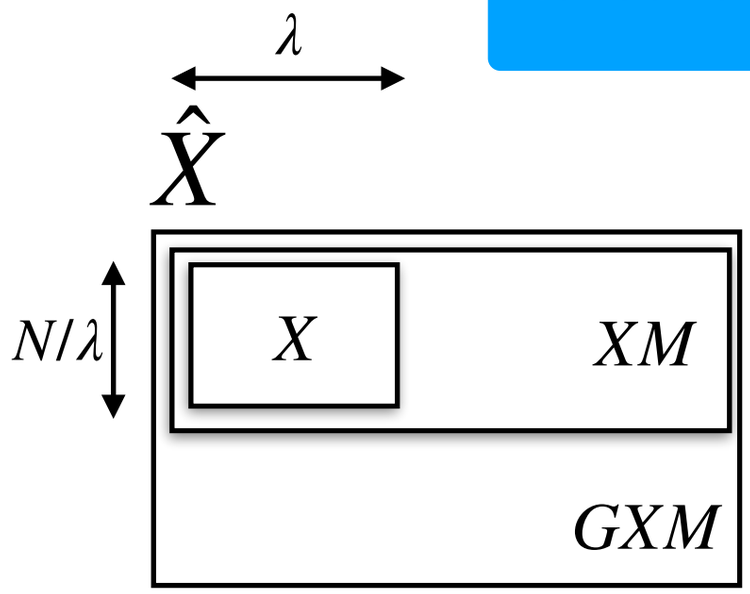
Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.



Pointwise Multiplication IOP

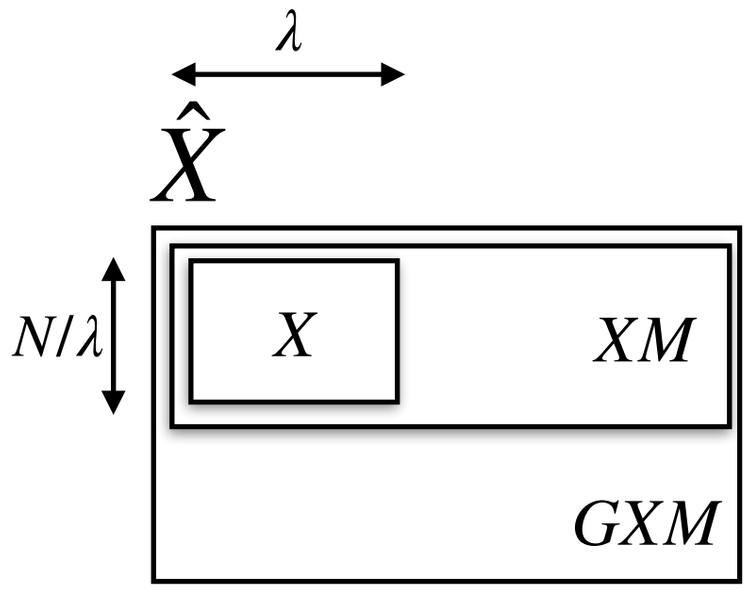
Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

tensor code!



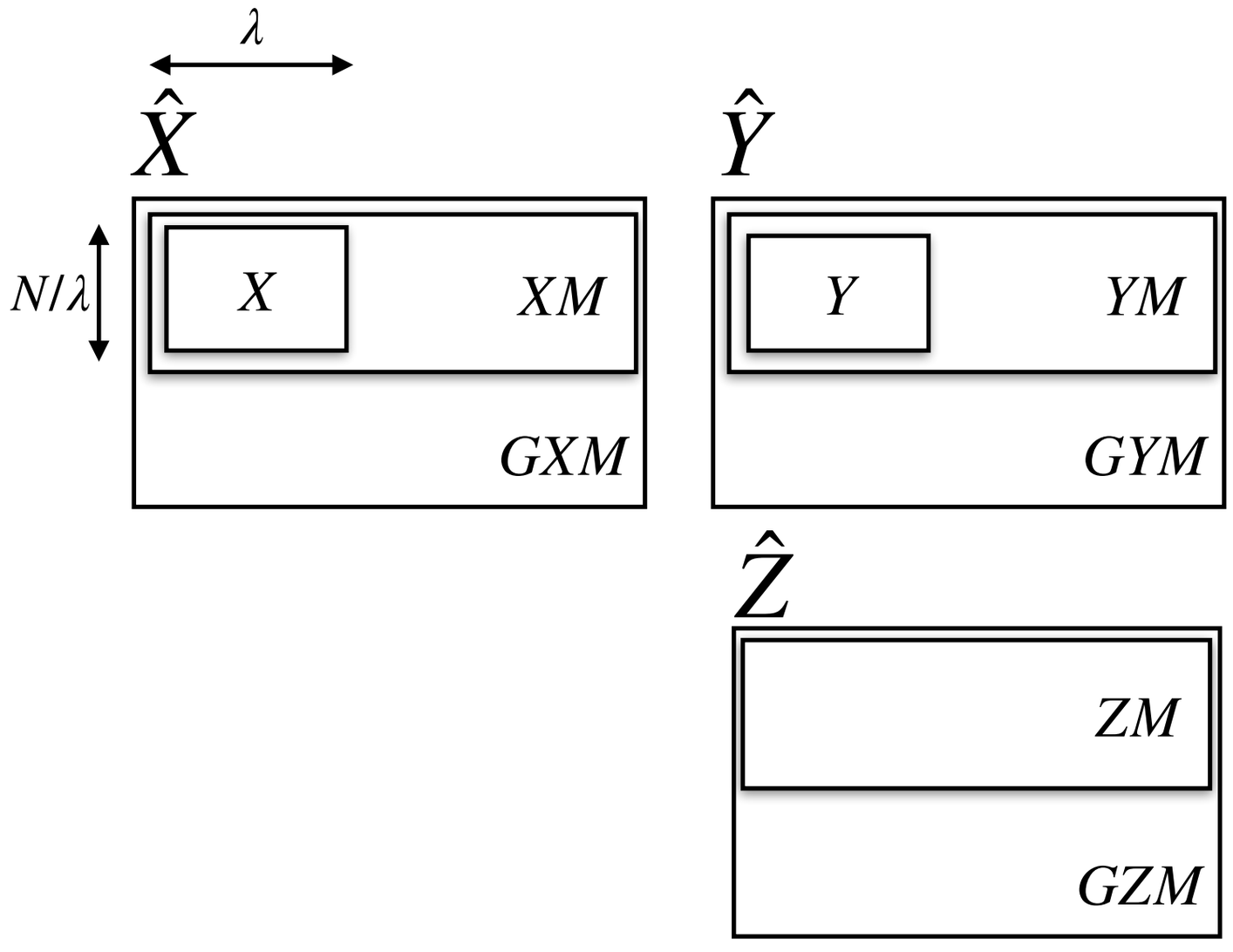
Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.



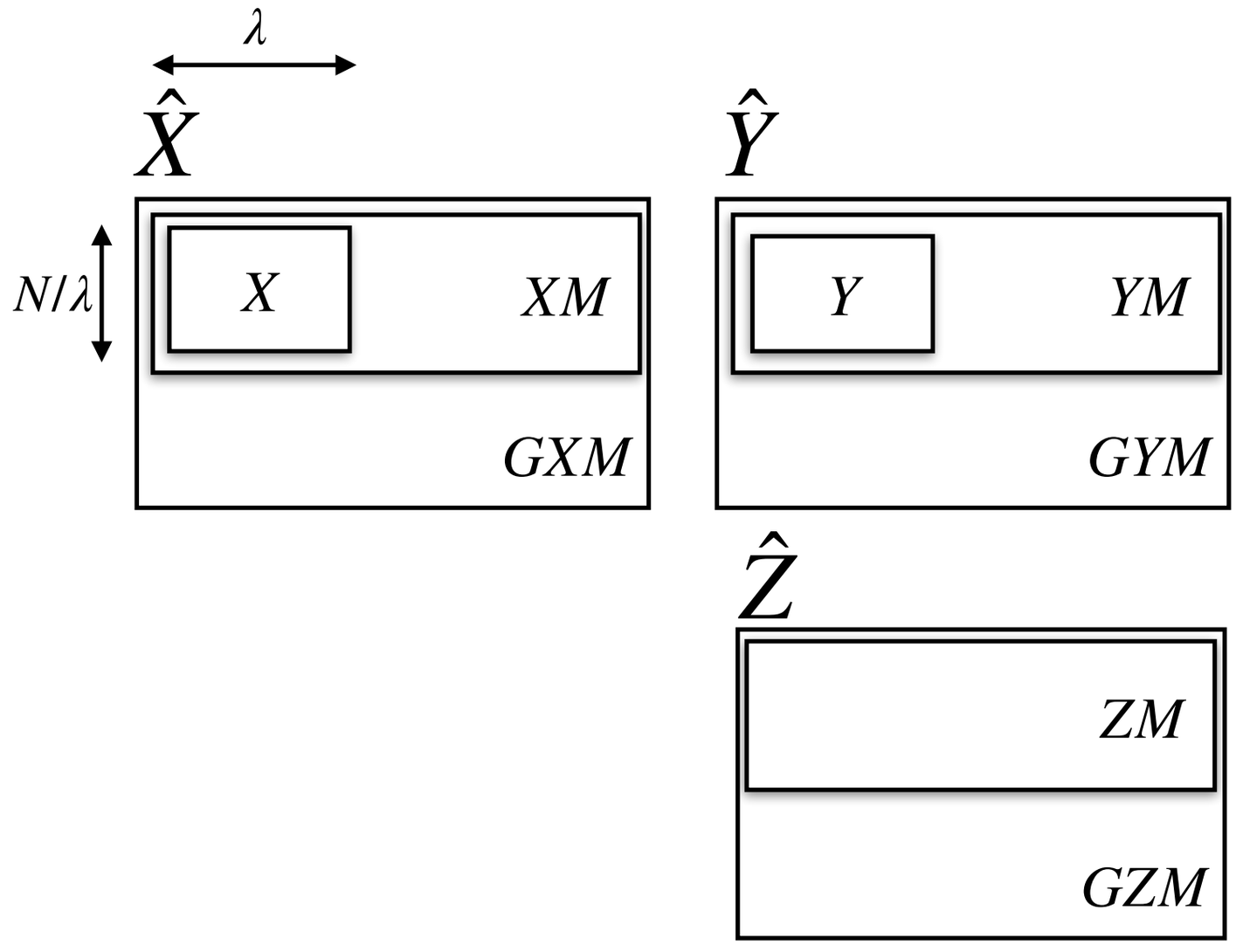
Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.



Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

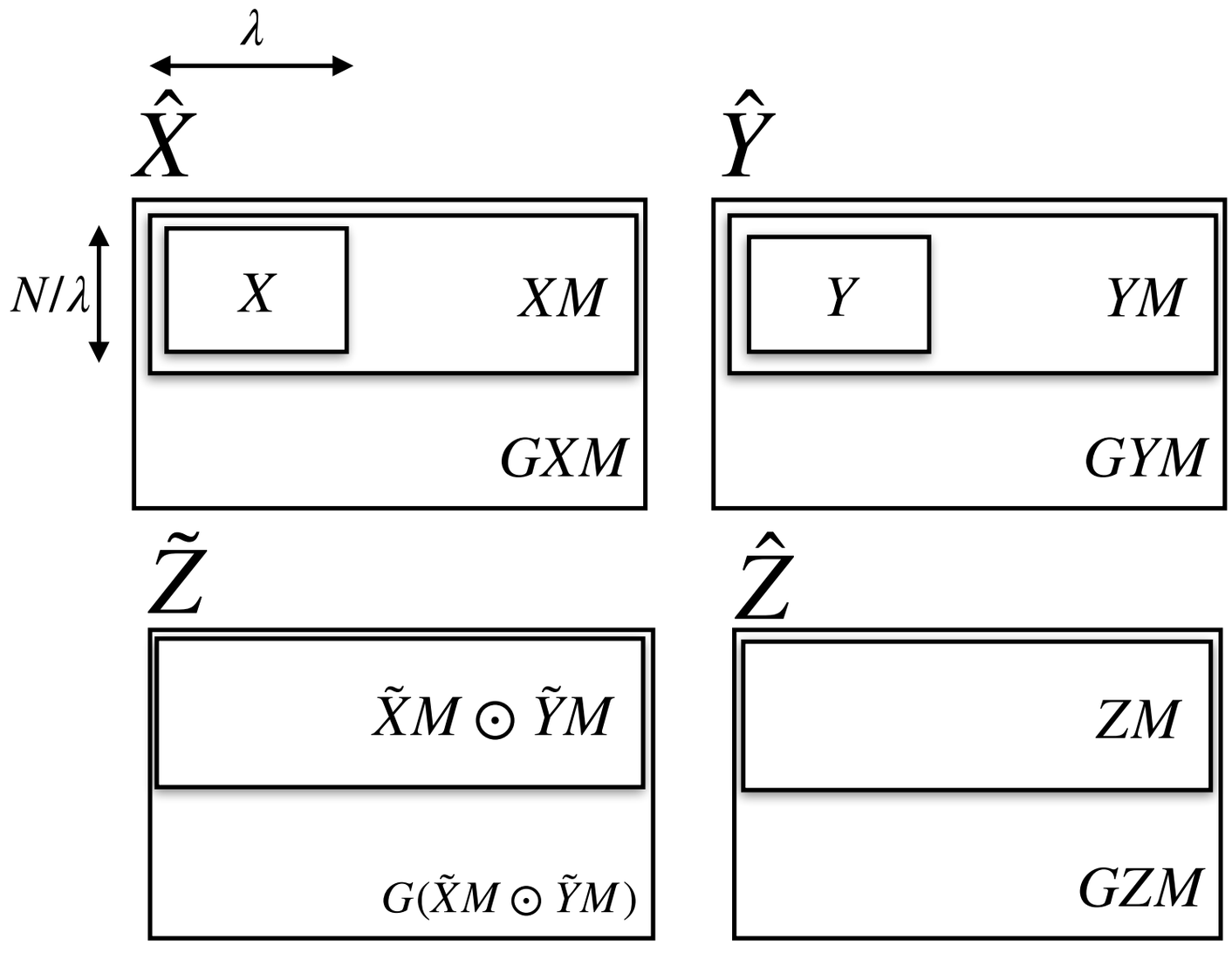


NOT TO SCALE!
matrices skinnier than shown

Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

1. Prover sends $\tilde{Z} = G(XM \odot YM)$



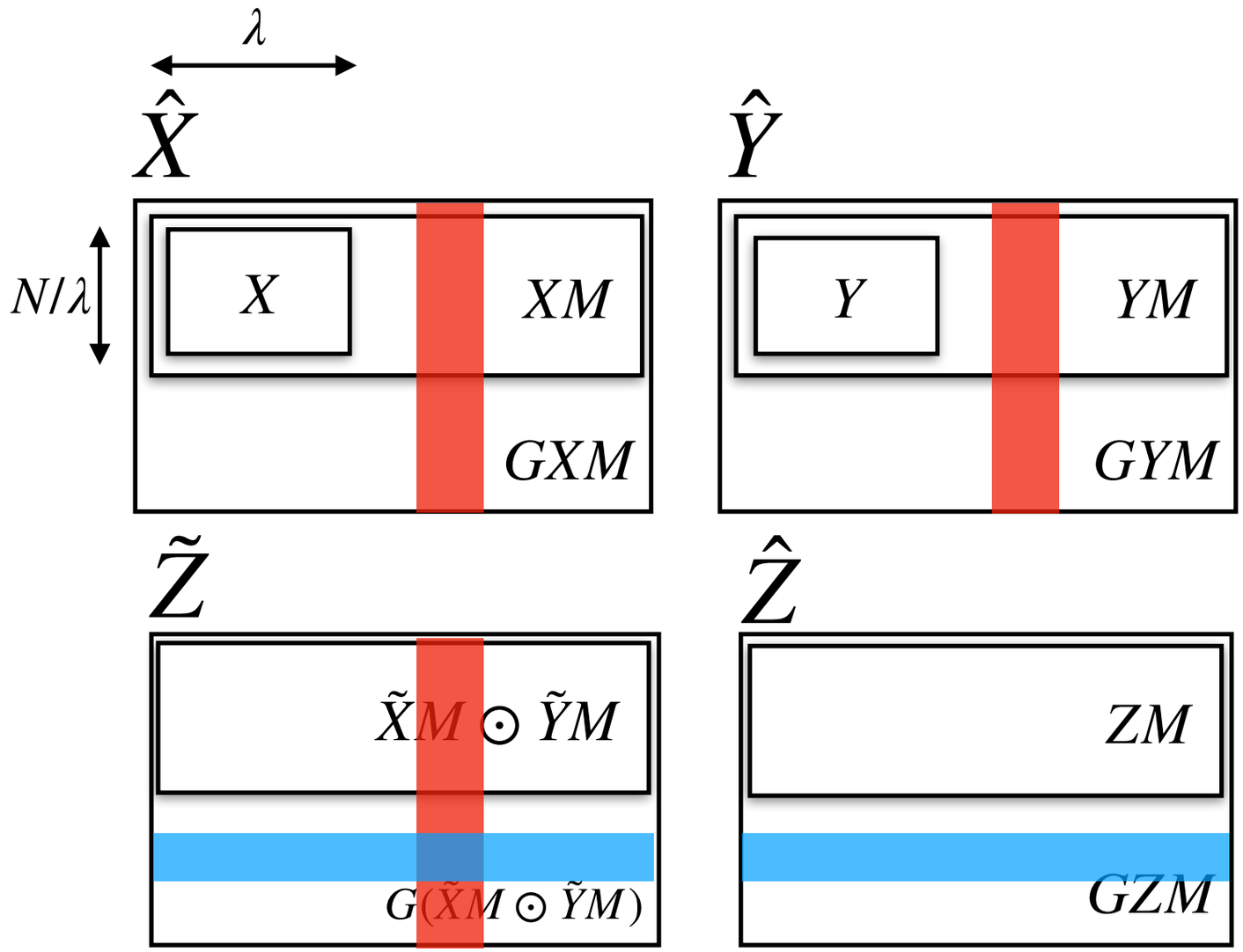
NOT TO SCALE!
matrices skinnier than shown

Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

1. Prover sends $\tilde{Z} = G(XM \odot YM)$
2. Verifier:

a) samples a column and row,
verifies (constant soundness):



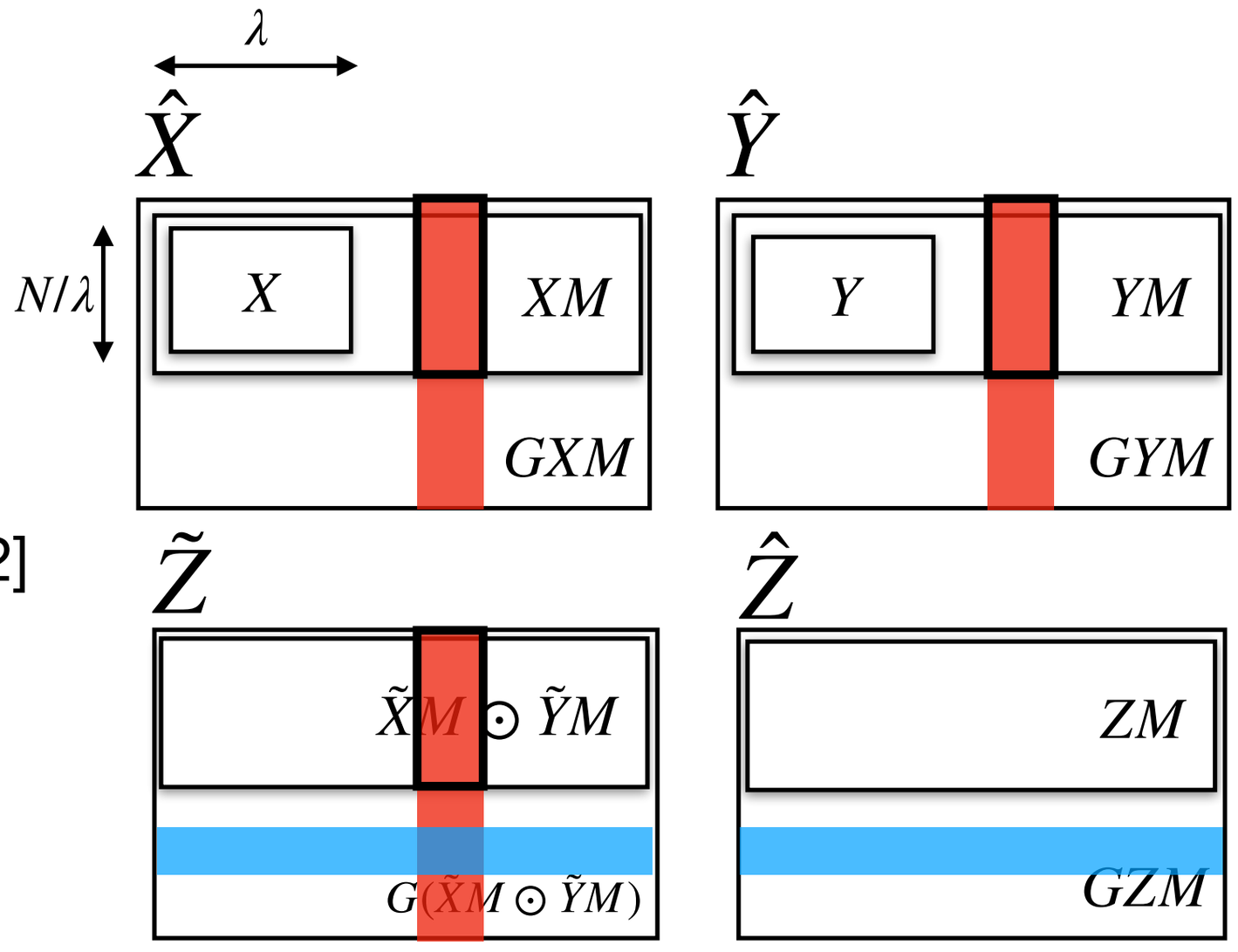
NOT TO SCALE!
matrices skinnier than shown

Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

1. Prover sends $\tilde{Z} = G(XM \odot YM)$
2. Verifier:
   a) samples a column and row, verifies (constant soundness):
      - column consistency via [RR22]

$$\implies \tilde{X} \odot \tilde{Y} = X \odot Y$$



NOT TO SCALE!
matrices skinnier than shown

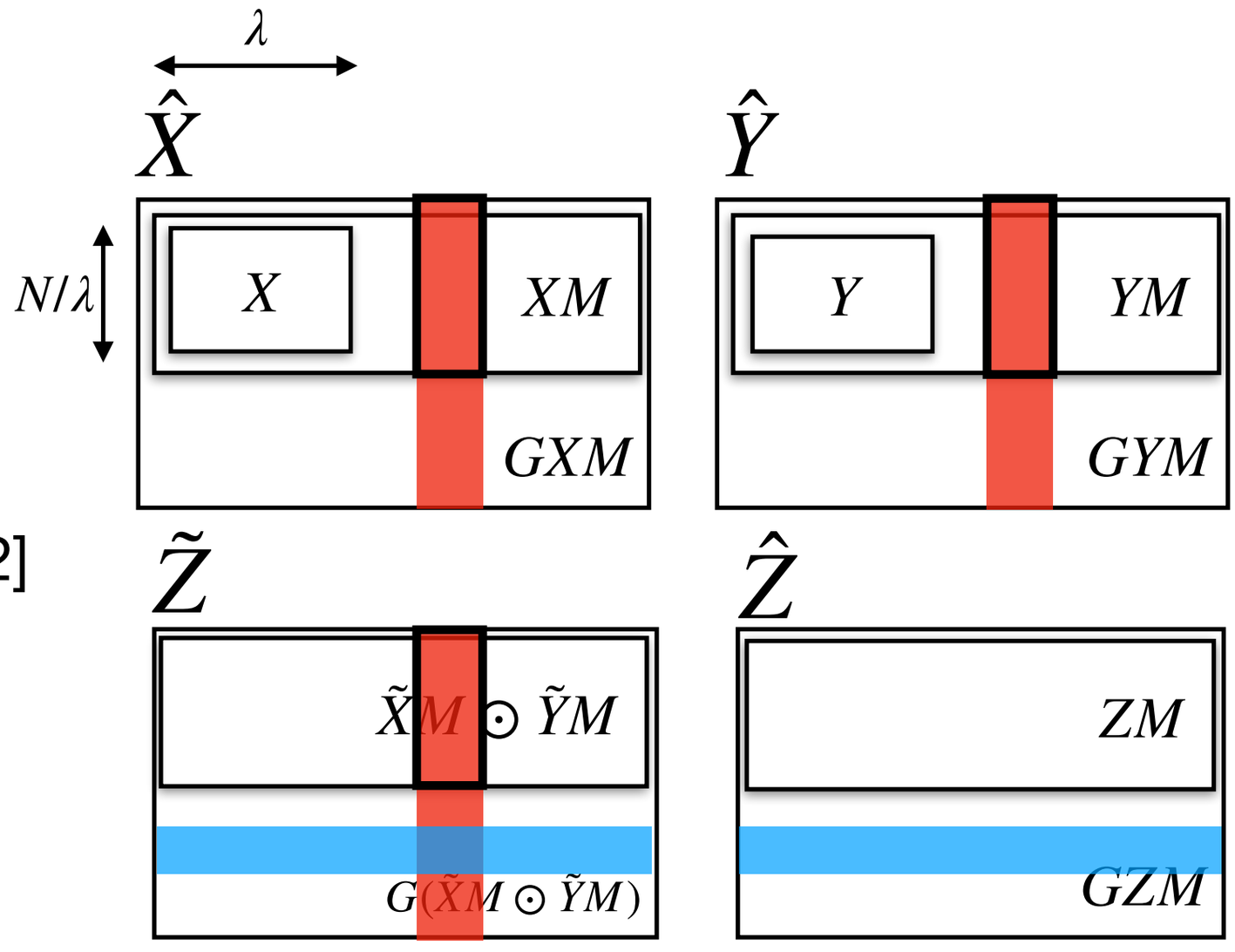
Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

1. Prover sends $\tilde{Z} = G(XM \odot YM)$
2. Verifier:
   a) samples a column and row, verifies (constant soundness):
      - column consistency via [RR22]

        $\implies \tilde{X} \odot \tilde{Y} = X \odot Y$
      - row consistency directly

        $\implies Z = \tilde{X} \odot \tilde{Y}$



NOT TO SCALE!
matrices skinnier than shown

Pointwise Multiplication IOP

Given encodings $\hat{X} = GXM$, $\hat{Y} = GYM$, and $\hat{Z} = GZM$,
want to establish: $Z = X \odot Y$.

1. Prover sends $\tilde{Z} = G(XM \odot YM)$
2. Verifier:

a) samples a column and row, verifies (constant soundness):

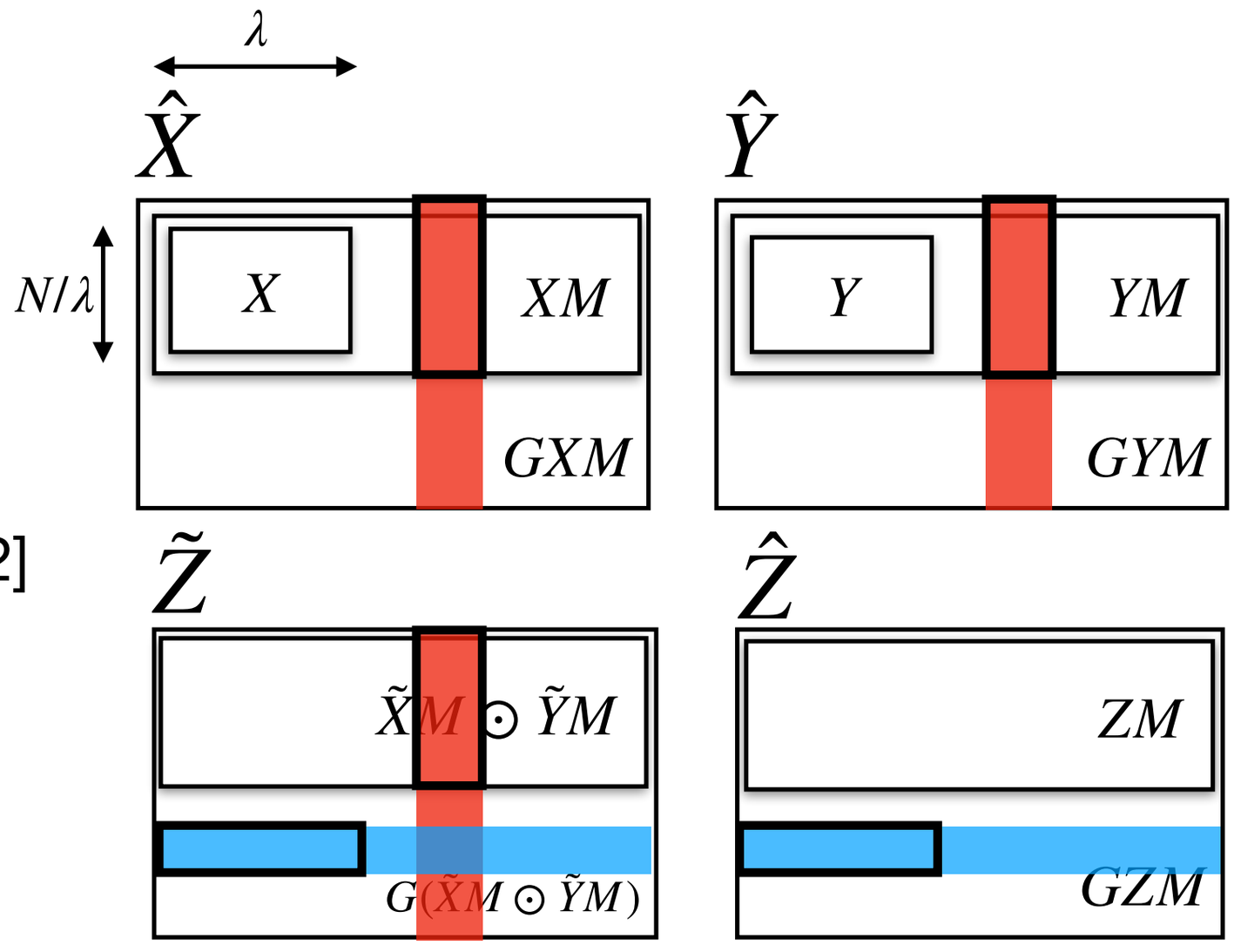
- column consistency via [RR22]

$$\implies \tilde{X} \odot \tilde{Y} = X \odot Y$$

- row consistency directly

$$\implies Z = \tilde{X} \odot \tilde{Y}$$

b) repeats λ times



NOT TO SCALE!
matrices skinnier than shown

Epilogue

Sequel?

Sequel?

- Reduce prover size to $O(S)$ + poly$(\lambda, \log S)$?

Sequel?

- Reduce prover size to $O(S)$ + poly$(\lambda, \log S)$?
  - Replace Reed-Solomon by linear-size encodable multiplication codes (do these exist?)

Sequel?

- Reduce prover size to $O(S)$ + poly$(\lambda, \log S)$?
  - Replace Reed-Solomon by linear-size encodable multiplication codes (do these exist?)
- *Non-interactive* argument for P with similar efficiency (currently from random oracle)

Sequel?

- Reduce prover size to $O(S)$ + poly$(\lambda, \log S)$?
  - Replace Reed-Solomon by linear-size encodable multiplication codes (do these exist?)
- *Non-interactive* argument for P with similar efficiency (currently from random oracle)
- Handle arbitrary Boolean circuits? (with preprocessing)

Sequel?

- Reduce prover size to $O(S)$ + poly$(\lambda, \log S)$?
  - Replace Reed-Solomon by linear-size encodable multiplication codes (do these exist?)
- *Non-interactive* argument for P with similar efficiency (currently from random oracle)
- Handle arbitrary Boolean circuits? (with preprocessing)

Paper

Slides