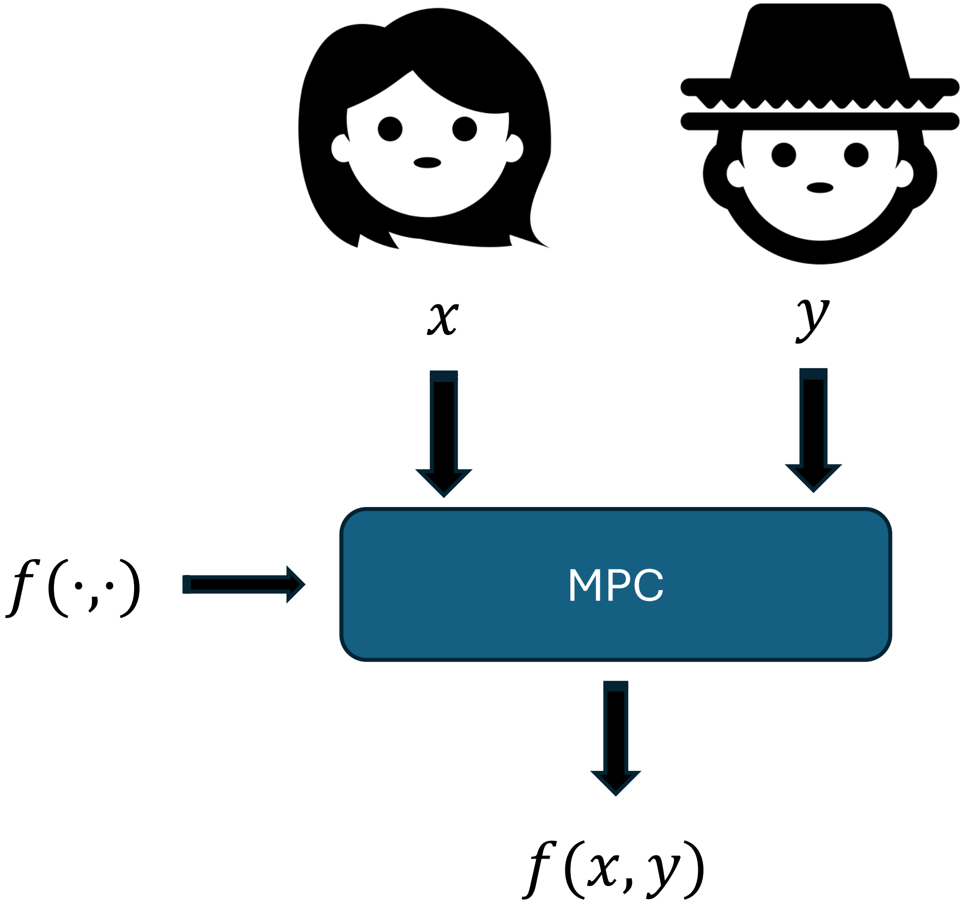
# Toss: Garbled PIR from Table-Only Stacking

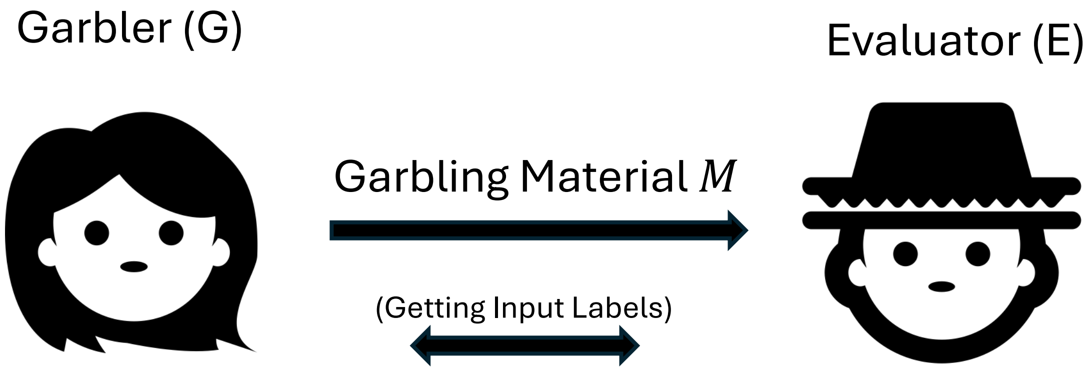
**Lucien K. L. Ng** and Vladimir Kolesnikov

Georgia Institute of Technology

# Multi-Party Computation (MPC)



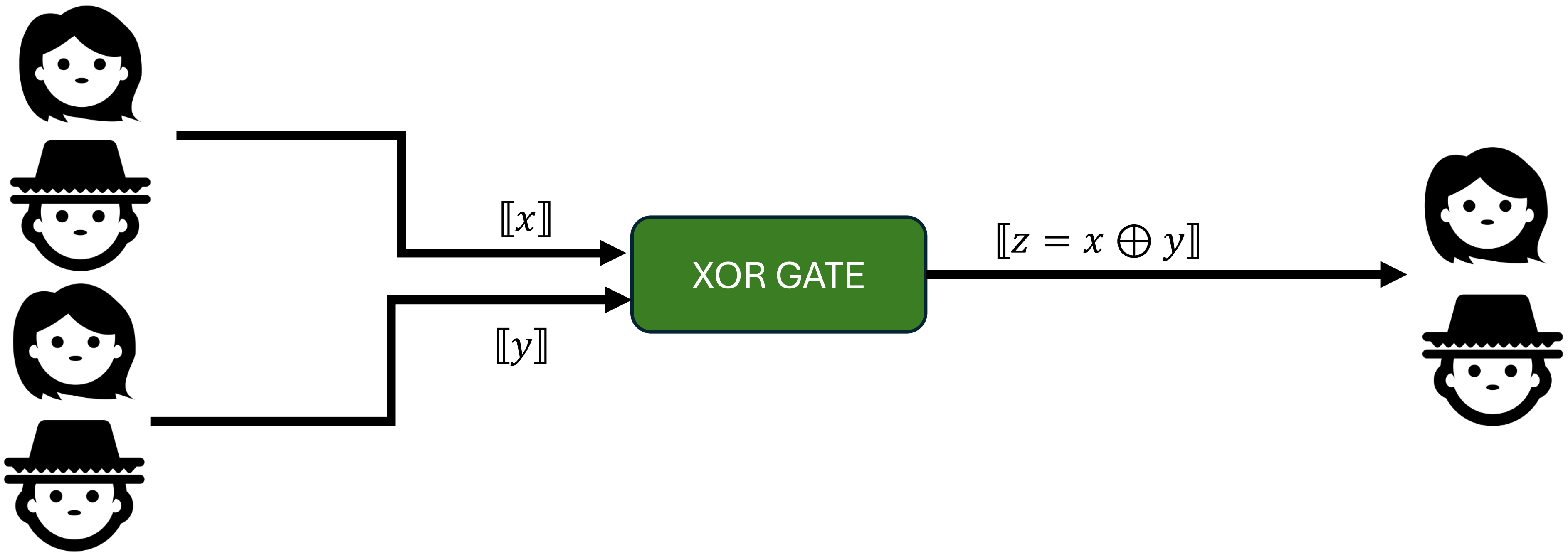
# Garbled Circuit (GC)




- 😄 Constant Rounds
- 😄 Symmetric Key Primitives
- 😄 Authenticity
- 😭 Limited Choice of Gates

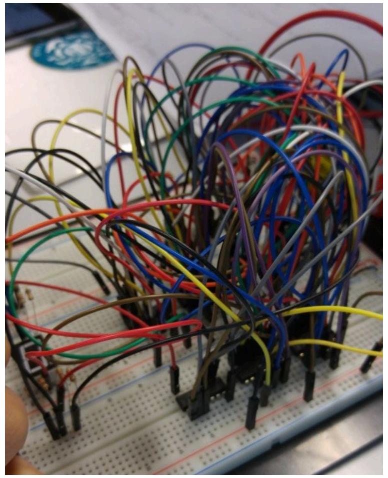
# Circuits as a Composition of Gates

- XOR gates: $XOR(x, y) = x \oplus y$
- AND gates: $AND(x, y) = x \cdot y$
- $x, y \in \{0, 1\}$

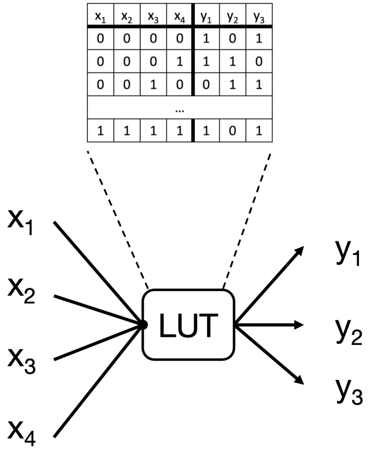


# How about Garbled Lookup Table (LUT)?

- Parameters: Lookup Table $\mathrm{F}$ of size $N = 2^n$
- Input: G and E hold garbled share index $[\![x]\!]$
  - $x$ may have $n \geq 1$ bit
- Output: G and E receive $[\![\mathrm{F}[x]]\!]$



vs.



# GLUT Applications

- Privacy-Preserving Machine Learning (PPML)
  - LUTs are used to compute highly-complicated functions and operations
  - SiRnn [SP'21] uses 1020-row LUTs for $1/\sqrt{x}$
  - Beacon [USS'23] uses $2^{12}$-row LUTs for tanh and sigmoid over half-floats
  - Sigma [PETS'24] uses $2^{13}$- and $2^{16}$-row LUTs for $1/x$
- Client-Malicious PPML due to GC's authenticity
  - MUSE [USS'21], SIMC [USS'22]
- Privacy-Preserving Deterministic Finite Automata
  - Substring Search (KMP Algorithm)
  - DNA Pattern Matching
- New Opportunity to Optimize Circuits

# Prior Art of Garbled LUT/PIR

- For now, we focus on communication cost
- Naive (Yao's [FOCS'86]): $2^n m\kappa$ bits
- logrow's [EC'24] : $\approx nm\kappa + \boxed{2^n m}$ bits
  - Security Parameter $\kappa$ is usually 128

no $\kappa$
but as large as the table

$n$ = # input bits
$m$ = # output bits

*Can we reduce the communication cost to be sublinear in table size and $\kappa$?*

logrow: David Heath, Vladimir Kolesnikov, **Lucien K. L. Ng**. Garbled Circuit Lookup Tables with Logarithmic Number of Ciphertexts. EC'24

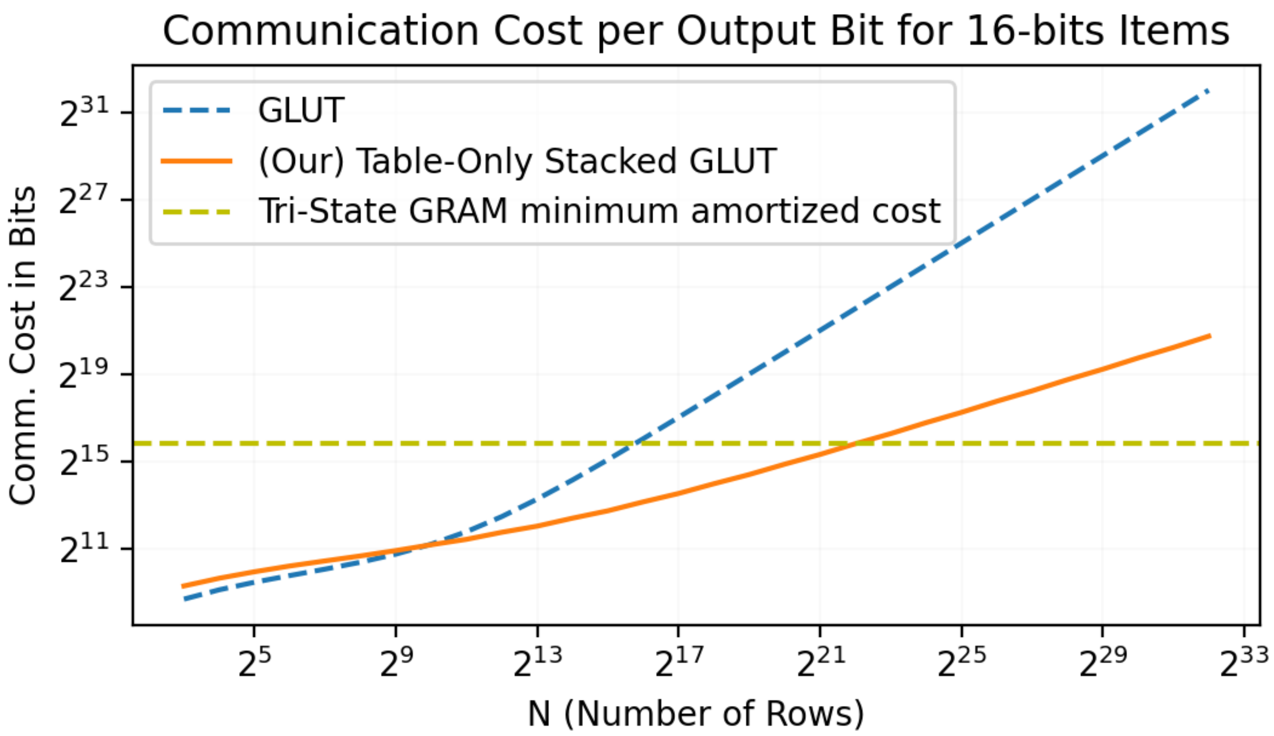
# Recent Efforts to Avoid $\kappa$

| Scheme | Optimization Target | Crypto Assumption |
| --- | --- | --- |
| **Ours** | LUT | 😄 (**Symmetric**) CCRH |
| BitGC [EC'25] | AND | (Asymmetric) RLWE/NTRU |
| Silent Circuit Relinearisation [Crypto'25] | AND/Mult/LUT | (Asymmetric) DCR |
| Breaking the 1/λ-Rate Barrier for Arithmetic Garbling [EC'25] | Mult | (Asymmetric) Power-DDH |
| $\omega(1/\lambda)$-Rate Boolean Garbling Scheme from Generic Groups [EC'25] | Mult | (Asymmetric) Generic Group model |
| Rate-1 Arithmetic Garbling From Homomorphic Secret Sharing [TCC'24] | Mult | (Asymmetric) DCR |
| New Ways to Garble Arithmetic Circuits [EC'25] | Mult | (Asymmetric) DCR/LWE |

- XOR or Addition are ignored as they are communication-free
- Mult: Arithmetic Multiplication Gate
- AND: Boolean AND Gate

# Toss' (Ours) Communication Cost

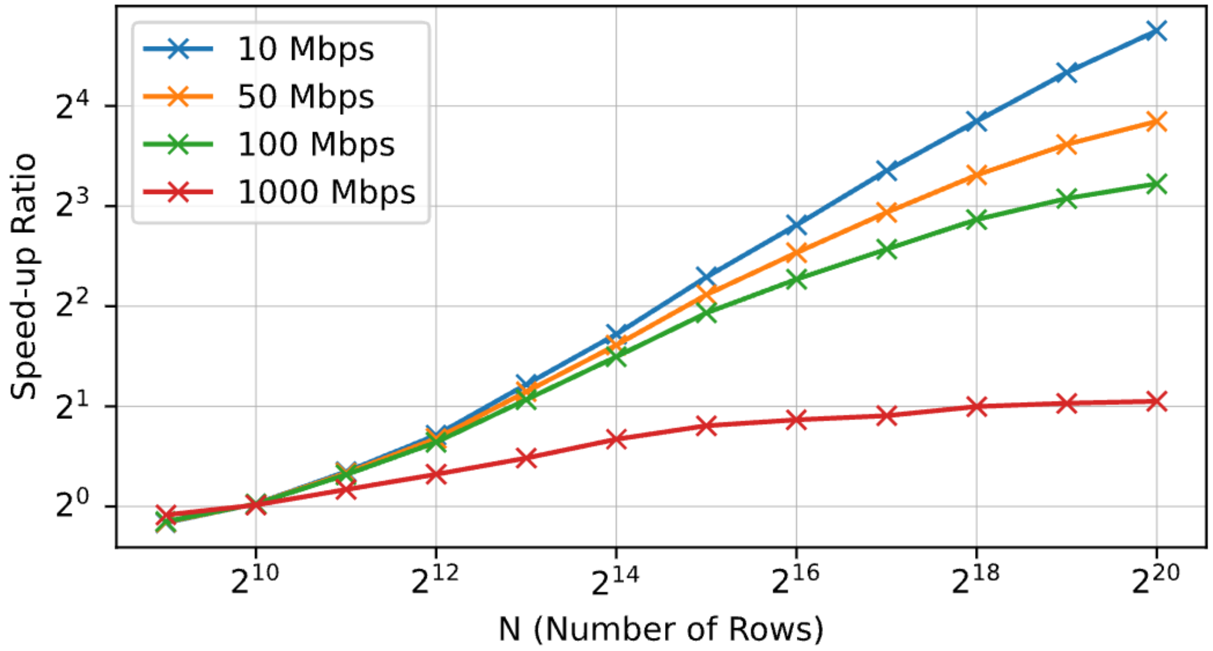
- Toss reduces the comm. cost to $O(\sqrt{N} m \sqrt{\kappa})$ bits
  - with a lean hidden constant ($< 10$)




Assume
- $m = 16$ bits items
- $\kappa = 128$

# Concrete Runtime Improvement

- Throughput increases from ≈10.6 lookup/s to ≈81 lookup/s (>7.5x)
  - For $N = 2^{20}$ with Laptop (M3 Pro Macbook) + 100Mbps Network




Assume
- $m = 16$ bits items
- $\kappa = 128$

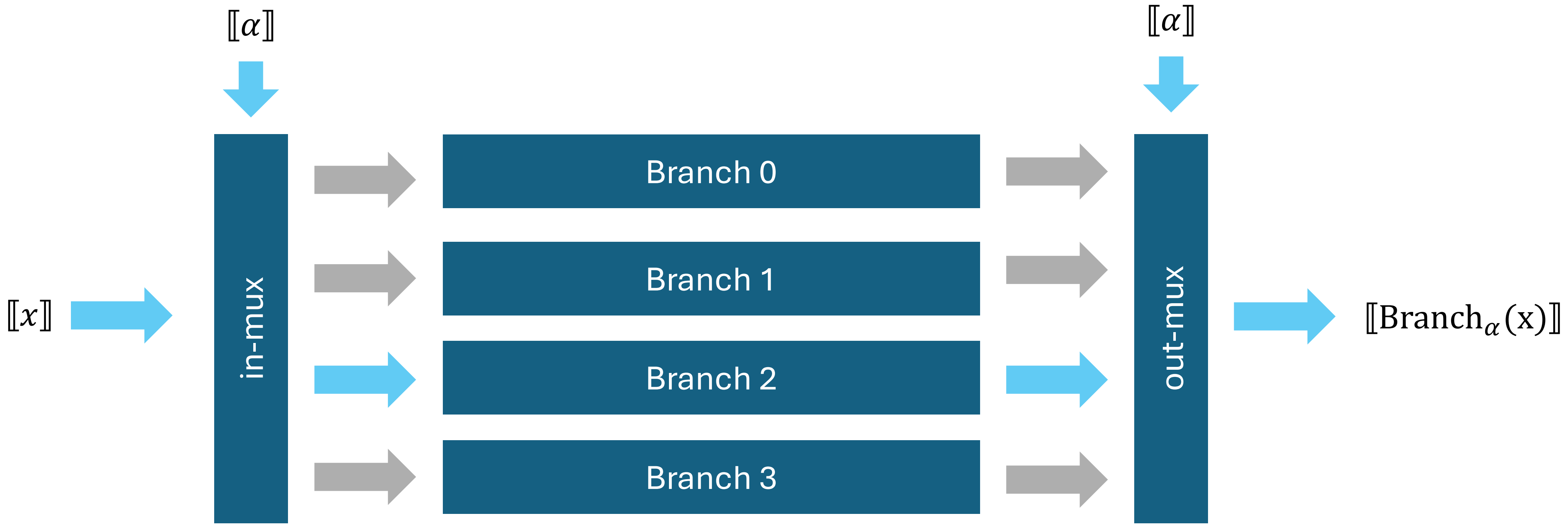
# Garbled PIR vs. LUT

- GPIR : Garbled Private Information Retrieval

| | GPIR | GLUT |
|---|---|---|
| Table Content | Known by both G & E (i.e., **Public**) | Known by G |
| Table Size | **Large** (e.g., N > 2^12) | Small |
| Scheme | **Toss (Ours)** | logrow<br>Yao's |

# Ingredient: Stacked Garbling [Crypto'20]

## Goal: Reduce communication cost for conditional programs



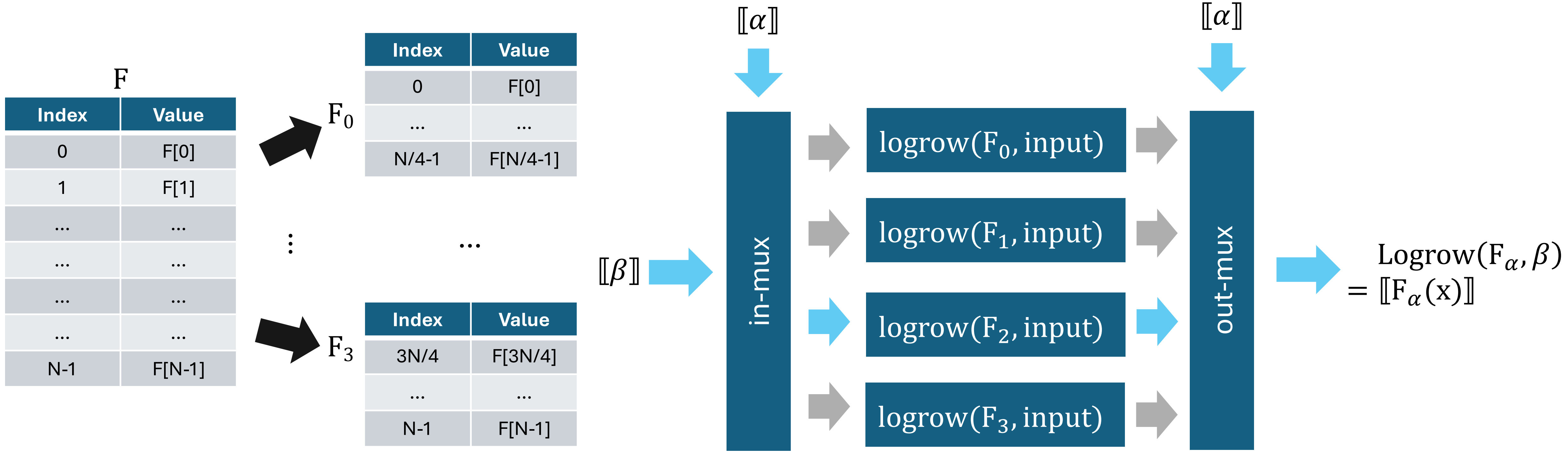

Communication Cost: $\underbrace{\text{Cost(in-mux)} + \text{Cost(out-mux)}}_{O(\#\text{Branch} \cdot (|\text{input}| + |\text{output}|) \cdot \kappa)}$ + $\text{Max}_i$ Cost($\text{Branch}_i$)

⟦$\alpha$⟧ is the garbled share of x

*David Heath and Vladimir Kolesnikov*. Stacked Garbling: Garbled Circuit Proportional to Longest Execution Path. Crypto 2020

# Basic Idea: Stacking logrow

- Decompose the lookup index $[\![x]\!] = [\![\alpha]\!] || [\![\beta]\!]$



# Communication Cost of Stacked logrow

$$\underbrace{\text{Cost(in-mux)} + \text{Cost(out-mux)}}_{O(B \cdot m \cdot \kappa)} + \underbrace{\text{Max}_i \ \text{Cost}(\text{Branch}_i)}_{\text{Cost}\left(\text{logrow}(\text{F}_i, \cdot)\right) = \log\frac{N}{B} m\kappa + \frac{N}{B} m}$$

Plugging in $B = \sqrt{N/\kappa}$ ➡ Cost = $O(\sqrt{N} m \sqrt{\kappa})$

- $B$ = # Branch/Sub-table
- $m$ = item size
- $N = 2^n$ = table size

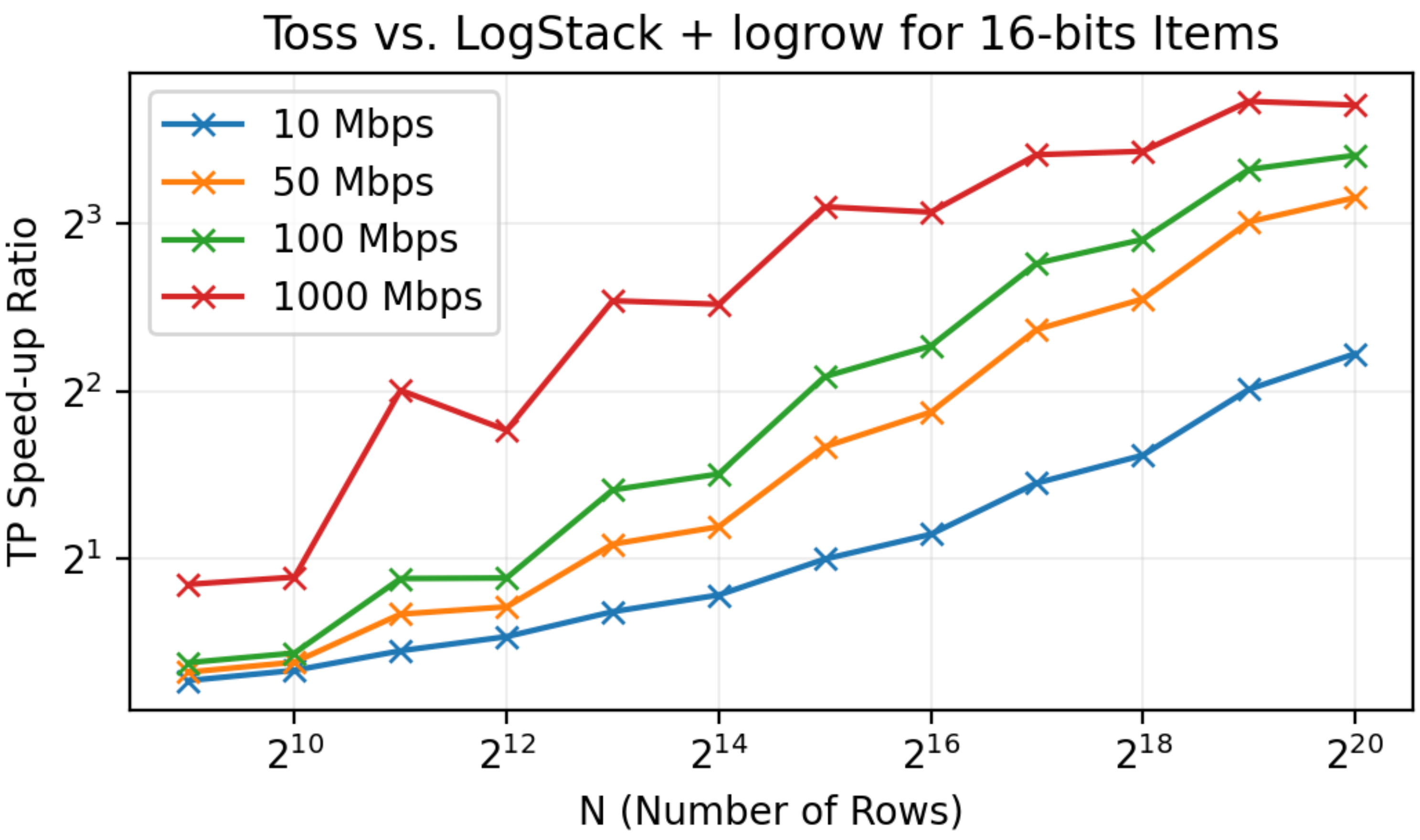
# …What about Computation?

- Original Stacked Garbling [Crypto'20]: $O(B^2 \cdot |\mathrm{Branch}|)$
- LogStack [EC'21]: $O(B \log B \cdot |\mathrm{Branch}|)$

- For our stacked logrow, the comp. cost becomes $O(\log \frac{N}{\kappa} \cdot Nm\kappa)$
  - vs. unstacked logrow's $O(Nm\kappa)$
  - E.g., for $N = 2^{24}$, it is stacked logrow is 8.5x slower.

*Can we avoid the $O\left(log \frac{N}{\kappa}\right)$ computation overhead?*

# Stacked logrow vs. Toss



# Toss: Table-only Stacking of Sub-tables

- We stack only the sub-tables instead of the entire logrow
  - Much more light-weight!
  - Enabling more low-level algorithmic optimization
- Toss still has $O(\log B)$ overhead
  - But it's on the plain sub-stables instead of on sub-circuit with $\kappa$-blowup
  - Toss's Stacking: $O(\log B) \cdot O(Nm)$ vs. Naïve:$O(\log B) \cdot O(Nm\kappa)$

- (Still, the comp. cost is $O(Nm\kappa)$ due to multiplexing gadgets)

# Closing Remark of Toss

- What else in our paper
  - Detailed ideas of table-only stacking
  - How to reduce the cost for multiplexing in stacked garbling
- Summary:
  - Toss reduces communication without adding more computation
  - Part of the effort of avoid $\kappa$-factor in GC communication cost
    - Toss uses only symmetric-key primitive!
  - We use stacked garbling as an important building block

- Ad: Another paper @ Blockchain and Distributed Systems #2
  - Lite-PoT: Practical Powers-of-Tau Setup Ceremony