Efficient Mixed Garbling from Homomorphic Secret Sharing and GGM-tree

Jian Guo, Wenjie Nan

Nanyang Technological University

May 2, 2025

Overview

Overview

Garbled Circuits: Background and Recent Advances

Boolean Garbling (Yao)

- Many optimizations proposed over the years.
- Liu et al. (2025): *1-bit garbling* of boolean circuits.

Arithmetic Garbling (Applebaum, Ishai, Kushilevitz, 2011)

- Extends garbling to arithmetic operations.
- **Recent breakthroughs:**
  - Ball et al. (2023): *Constant-rate* garbling over bounded integers.
  - Meyer et al. (2024): *Rate-1* garbling over bounded integers.
  - Li & Liu (2024), Heath (2024), Ishai et al. (2025): *Garbling over mixed circuits* (bounded integers, integer rings, and boolean circuits)

Our Problem: Garble Mixed Circuits

Limitations of Prior Work:

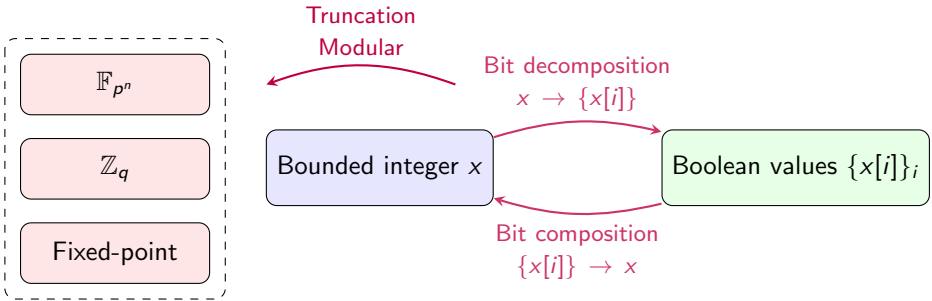
- **Incompatibility:** Not compatible with state-of-the-art *constant-rate* garbling over bounded integers.
- **Efficiency:** For $\mathbb{Z}_{q\sim 2^b}$ (integer rings):
  - Asymptotic $O(\lambda b)$ via CRT, **or**
  - $O(\lambda_{\mathrm{DCR}} b)$ via public-key schemes.
- **Narrow Domains:** Lack support for:
  - Finite fields $\mathbb{F}_{p^n}$,
  - Fixed-point numbers (approximating $\mathbb{R}$ with precision loss).

Our Goal:

Design clean protocols with improved concrete efficiency for mixed garbling across various domains.

Core Innovation

Choice of base domain: Binary circuits + bounded integer arithmetic



Innovation: Efficient bit decomposition/composition:

- Compatible with state-of-the-art arithmetic garbling.
- Useful for garbling over other domains.

Our Results

Parameters: $\lambda_{\mathsf{DCR}} \approx 3096$, $\lambda \approx 128$ (k is a trade-off parameter with $O(2^k)$ computable)

Key Improvements

- **Bit Decomposition/Composition on Signed Integers** ($(-2^{b-1}, 2^{b-1})$): Improved from $O(\lambda_{\mathsf{DCR}} b)$ to $O(\frac{\lambda_{\mathsf{DCR}}}{k} b)$. When $k = 8$, we can achieve $9\lambda b, b \sim 100 \to 6\lambda b, b \sim 1000$. Previous best results is $\geq 50\lambda b$ [LL23].
  - **Integer Rings** ($\mathbb{Z}_{2^b}$): Addition/Multiplication: $\sim$ bit decomposition, where addition can be made free in practice.
  - **Fixed-Point Arithmetic**: Free addition, multiplication: $O(\frac{f}{k}\lambda_{\mathsf{DCR}})$ (precision 2^{-f}).
  - **Finite Fields** ($\mathbb{F}_{p^n}$): $O(\frac{\lambda_{\mathsf{DCR}}}{k} b)$, $b = n\lceil \log p \rceil$ vs. naive $O(n^2 \log p)$.
- **Bounded Integers** (vs. Meyer et al.): $\frac{\gamma+1}{\gamma-2} \to \frac{\gamma}{\gamma-2}$.

Overview

Challenges in Bit Decomposition

Current Approach: Mod-and-Reduce Method

- Requires long arithmetic labels: $\vec{\Delta}_Z x + \vec{K}_Z$
- Extracting the LSB (Least Significant Bit) involves:

$$\vec{\Delta}_Z x + \vec{K}_Z \bmod 2 = \left(\Delta_{Z,1} x + K_{Z,1} \bmod 2, \ldots, \Delta_{Z,\lambda} x + K_{Z,\lambda} \bmod 2\right)$$

- Problem: Each arithmetic label contains $\geq \lambda$ values

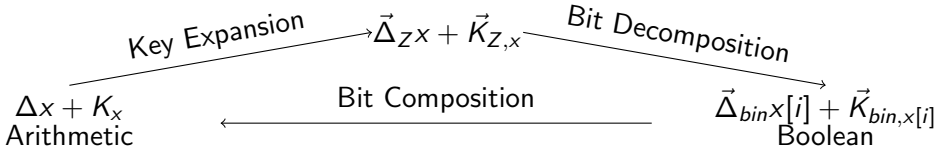
State-of-the-Art Arithmetic Garbling (MORS24)

- Uses *compact* labels: Just **one** value per arithmetic label
- Key Question: How to perform efficient bit decomposition with these shorter labels?

New Method for Bit Decomposition/Composition

Core Idea:

- **Expansion** from short $(\Delta x + K_x)$ to long $(\vec{\Delta}_Z x + \vec{K}_{Z,x})$ arithmetic labels



Current Benefits:

- Bit composition already improved to $\sim \lambda_{\mathrm{DCR}} b$ via symmetric-key operations:

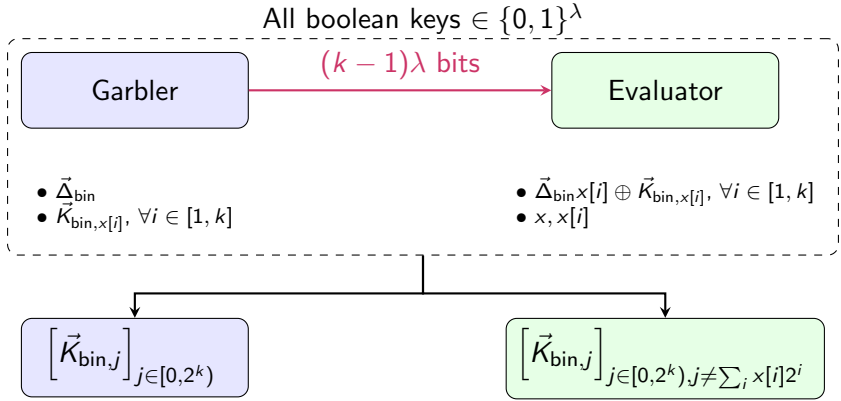
  $\mathrm{Enc}(sk, msg), \quad sk = \vec{\Delta}_{\mathrm{bin}} \cdot \{0,1\} \oplus \vec{K}_{\mathrm{bin},x[i]}, \quad msg = \Delta \cdot \{0,1\} + K_{x[i]}, |\Delta| \sim \lambda_{DCR}$

- Final result: $\sum_i (\Delta x[i] + K_{x[i]}) 2^i$

Can we achieve further improvements?

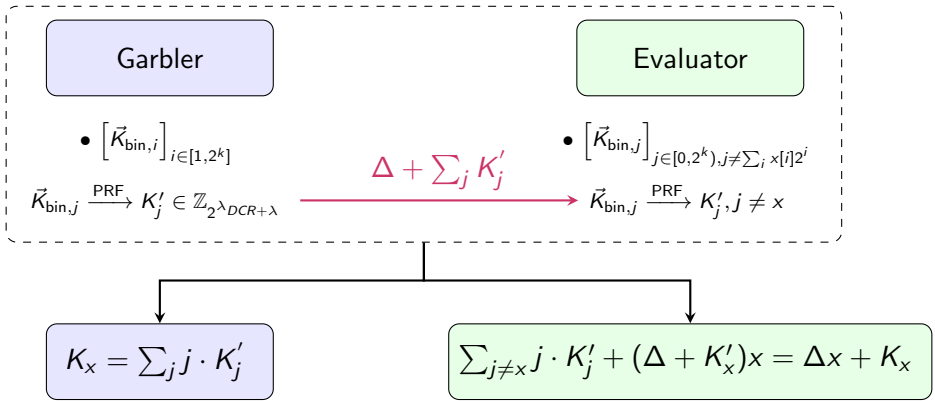
Overview

GGM-Tree - Half-Tree Protocol [Guo et al.23]



Computation cost: $O(2^k)$

Improved Bit Composition using GGM-Tree



Communication cost: $\lambda_{DCR} + \lambda$ per k bits (was per bit before)

Remaining Issues

Issues:

- Evaluator needs to know each bit $x[i]$
- Need to handle negative numbers

Solution: Masking Technique

1. Garbler picks random $r \in \mathbb{Z}_{2^{b+\lambda}}, r \gg |x|$.
2. Reveal $y = x + r > 0$ and $\vec{\Delta}_{bin} y[i] \oplus \vec{K}_{bin,y[i]}$ through addition circuits to evaluator.
3. Does bit composition on y instead of x.

Before:

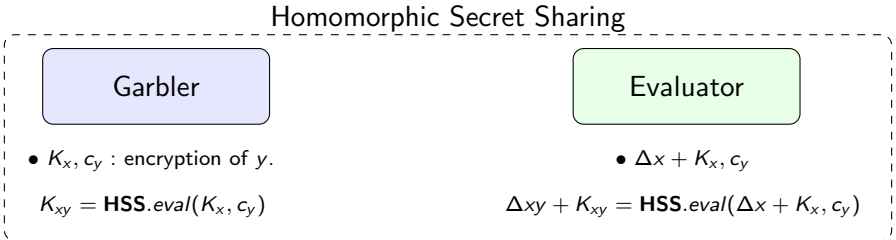
$$(K_y, \Delta y + K_y)$$

After:

$$(K_x = \Delta r + K_y, \Delta x + K_x = \Delta y + K_y)$$

Overview

"Free" Fixed Key Expansion



Goal: Convert short label $\Delta x + K_x$ to long label $\vec{\Delta}_Z x + \vec{K}_{Z,x}$

- random $\vec{K}_{Z,x} \gg \vec{\Delta}_Z x \rightarrow$ fixed $\vec{\Delta}_Z$ (global constant)

Garbler:

- Knows K_x, sends encryption of $\Delta^{-1}\vec{\Delta}_Z$

Evaluator:

- Knows $\Delta x + K_x$

$$\mathbf{HSS}.eval(c_{\Delta_{Z,i}}, K_x) = \vec{K}_{Z,x}[i], \mathbf{HSS}.eval(c_{\Delta_{Z,i}}, \Delta x + K_x) = \vec{\Delta}_Z[i]x + \vec{K}_{Z,x}[i]$$

Bit Decomposition via Fixed Key Expansion

Protocol Steps:

1. **Expand Labels:**
   - Short $\mathbf{L}(x) = \Delta x + K_x$
   - $\rightarrow$ Long $\vec{\Delta}_Z x + \vec{K}_{Z,x}$
2. **Garbler Computes:**
   - $sk_0 = \vec{K}_{Z,x} \bmod 2$
   - $sk_1 = (\vec{\Delta}_Z + \vec{K}_{Z,x}) \bmod 2$
3. **Encrypt/Transfer:**
   - Encrypt both cases (bit 0/1) of
     - $\vec{K}_{bin,x[1]} || K_{x[1]}$
     - $\vec{\Delta}_{bin} \oplus \vec{K}_{bin,x[1]} || \Delta + K_{x[1]}$

     using sk_0, sk_1 separately and send to evaluator.
   - Evaluator decrypts correct one

Iterative Process:

- Update for next bit:

$$K_x \leftarrow (K_x - K_{x[1]})/2$$

$$\mathbf{L}(x) \leftarrow \Delta \frac{x - x[1]}{2} + \frac{K_x - K_{x[1]}}{2}$$

- Repeat for all bits

Masking Enhancement

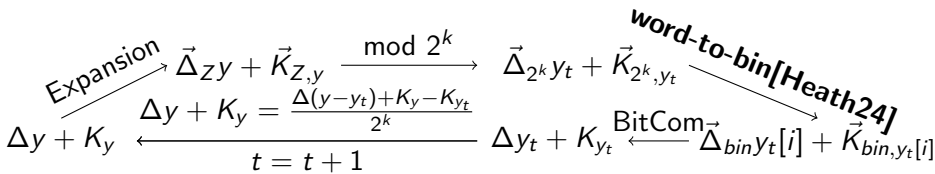
- Reveal $y = x + r$ instead of x
- Decompose y
- Subtract r via subtration circuit

Communication cost $\sim \lambda_{DCR} b$

First Improved Bit Decomposition

Heath24's Word-to-Bin Protocol

- Converts arithmetic labels over $\mathbb{Z}_{2^k}$ to boolean labels
- Communication cost: $2k\lambda$ per iteration



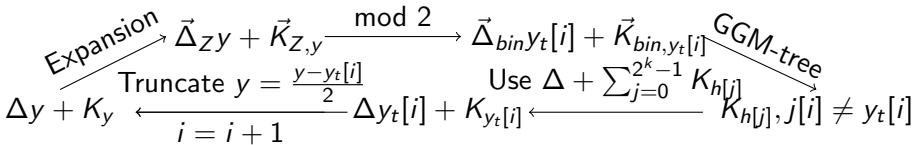
Can we reduce GGM-tree usage? Currently used twice

- In **word-to-bin** protocol and bit composition.
- Potential savings: $\sim 2b\lambda$ communication bits

Final Optimized Bit Decomposition Protocol

Key Innovation

- Single GGM-tree evaluation for every k bits
- "Free" fixed key expansion allows the truncation for each bit



- For each bit $y_t[i]$, parties obtain:
  - Garbler: $K_{y_t[i]}$
  - Evaluator: $\Delta y_t[i] + K_{y_t[i]}$
- Critical ingredient: Evaluator uses $\Delta + \sum_{j=0}^{2^k-1} K'_j$

Overview

Concrete Efficiency for BD/BC

Table: Efficiency (bits) of garbling schemes for $(-2^{b-1}, 2^{b-1})$ integers

Assumption	**Operation**	**General** (b, k)	**(128,8)**	**(1280,16)**
CCR-KDM	BitDecom	$(4b-2)\lambda + b + \frac{b}{k}\lambda_{\mathsf{DCR}}$	114,560	902,144
DPDL-3	BitDecom	$(2.5b-2)\lambda + b + \frac{b}{k}\lambda_{\mathsf{DPDL}}$	49,024	491,520
[BLLL23]	BitDecom	$\geq \lambda(b+\lambda_{\mathsf{DCR}})^2$	$\geq 1.3\times10^9$	$\geq 2.4\times10^9$
[LL23]	BitDecom	$2b\lambda_{\mathsf{DCR}} + 5b\lambda$	868,352	8,683,520
CCR-KDM	BitCom	$3(b\lambda+\lambda^2) + b + \frac{b}{k}\lambda_{\mathsf{DCR}}$	147,584	786,560
DPDL-3	BitCom	$2.5(b\lambda+\lambda^2) + b + \frac{b}{k}\lambda_{\mathsf{DPDL}}$	90,240	532,608
[LL23]	BitCom	$(2b+2)\lambda_{\mathsf{DCR}} + 6b\lambda$	890,880	8,853,504

- Blue rows: KDM security of Damgård-Jurik encryption in circular correlation robust hash function model.
- Green rows: Partial discrete logarithm assumption, use a smaller secret key $\Delta | (p-1)(q-1)$.

Concrete Efficiency for Other Domains

Table: Efficiency (bits) for Multiplication over $\mathbb{Z}_{2^b}$

Assumption	Gadget	General (b, k)	(128,8)	(1280,16)
CCR-KDM	MULT	$(\lceil \frac{2b+2\lambda}{\lambda_{\mathsf{DCR}}} \rceil + \frac{b}{k} + 4)\lambda_{\mathsf{DCR}} + 4b\lambda + 2b$	130,304	919,040
DPDL-3	MULT	$(\lceil \frac{2b+2\lambda+2\lambda_{\mathsf{DPDL}}}{\lambda_{\mathsf{DPDL}}} \rceil + \frac{b}{k} + 1)\lambda_{\mathsf{DPDL}} + 2.5b\lambda + 2b$	53,120	459,904
[LL23]	MULT	$\geq 4b\lambda_{\mathsf{DCR}} + 10b\lambda$	1,736,704	17,367,040
[BLLL23]	MULT	$\geq 48\lambda(\lambda_{\mathsf{DCR}} + b)$	19,660,800	26,738,688

Table: Efficiency for Fixed-Point Numbers in $(-2^{300}, 2^{300})$ with Precision 2^{-f}

Assumption	Gadget	General (f, k)	(64,8)	(128,16)
CCR-KDM	ADD	free	free	free
CCR-KDM	MULT	$2f\lambda + f + (4 + \frac{f}{k})\lambda_{\mathsf{DCR}}$	53,312	69,760

THANK YOU FOR LISTENING