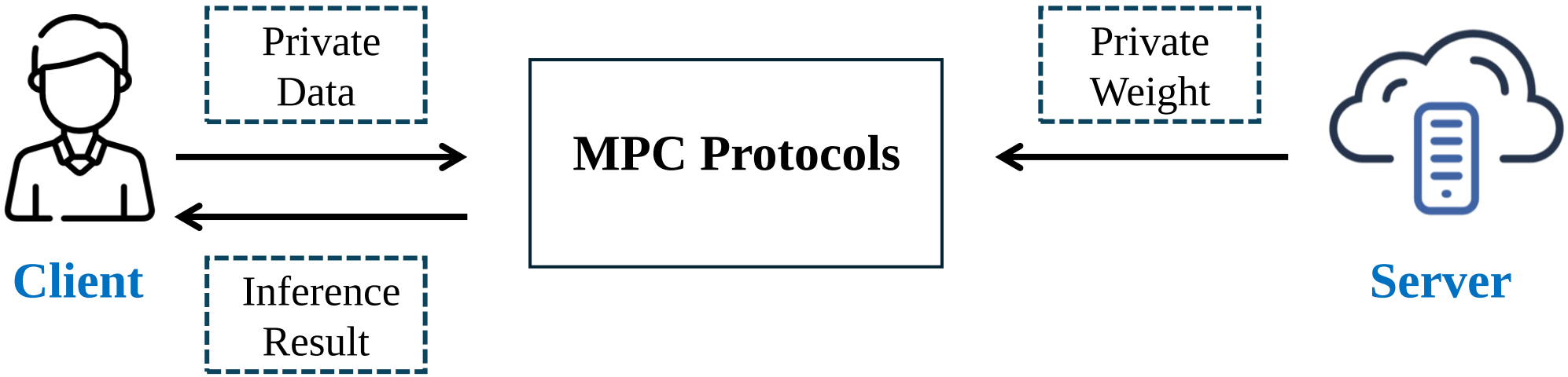
Distributed Function Secret Sharing and Applications

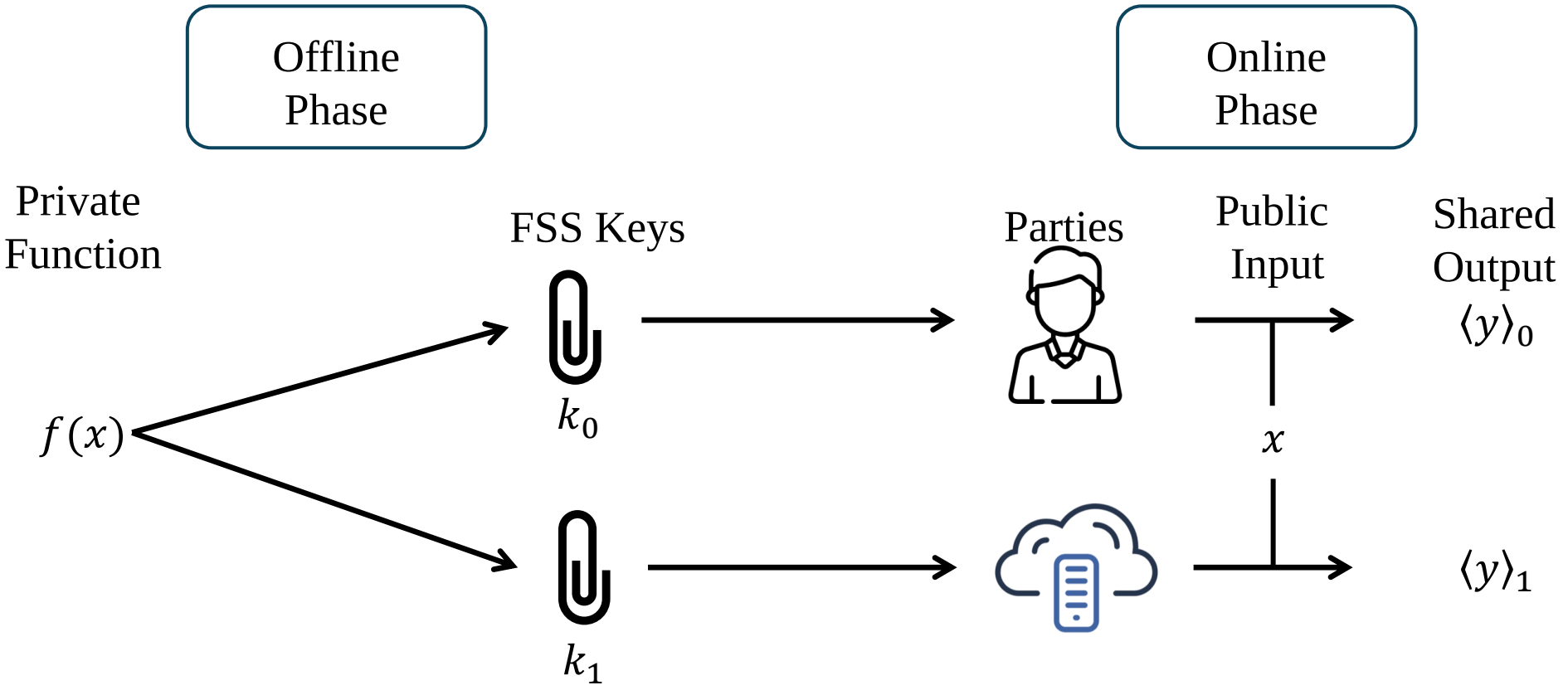
Pengzhi Xing, Hongwei Li, Meng Hao, Hanxiao Chen,
Jia Hu, Dongxiao Liu

Model Inference via Secure Multiparty Computation



- **Input:** Client secret shares the data & Server secret shares the weights.
- **Protocol Execution**: Communicate & compute in multiple rounds.
- **Output**: Client get the inference result.

Function Secret Sharing [BGI19]



- **Offline Stage**: Dealer generates FSS keys
- **Online Stage**: Parties jointly evaluate the key on the public input

Existing FSS is not Sufficient

Challenge 1:

Dealer-based FSS is unrealistic and weakens security guarantees.

- Finding a trusted party is difficult.
- A colluded dealer may compromise the privacy of honest party.

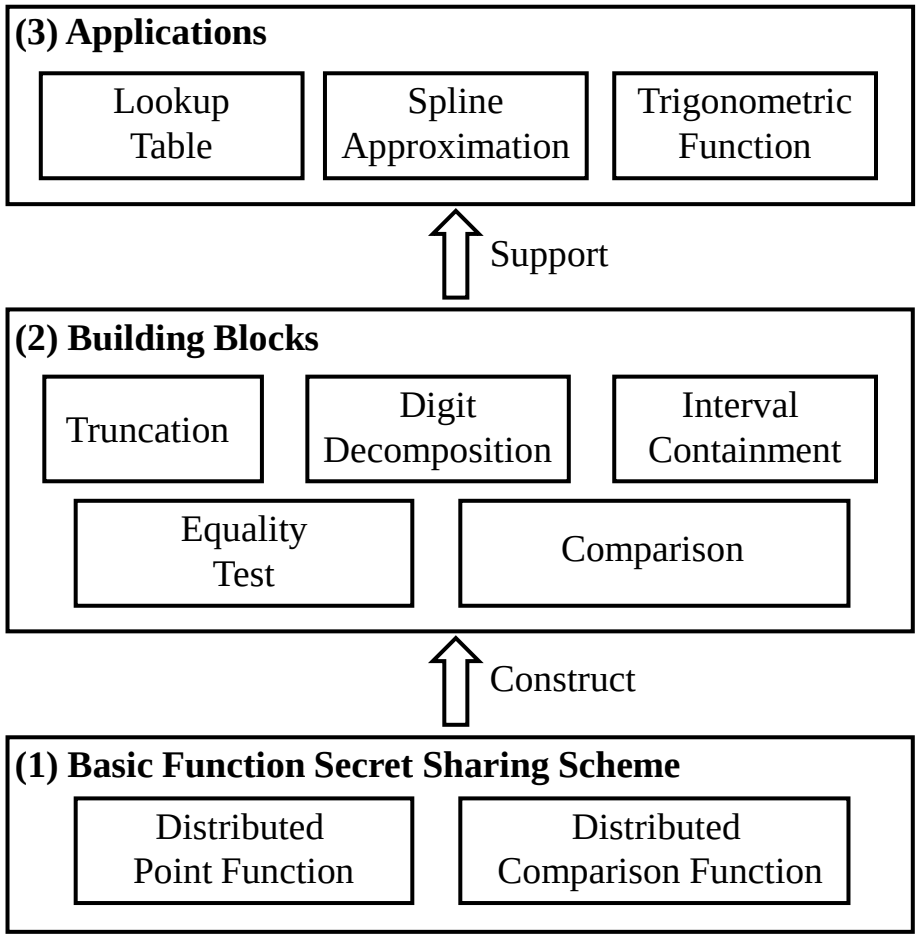
Challenge 2:

Existing dealer-less FSS have limited practicality.

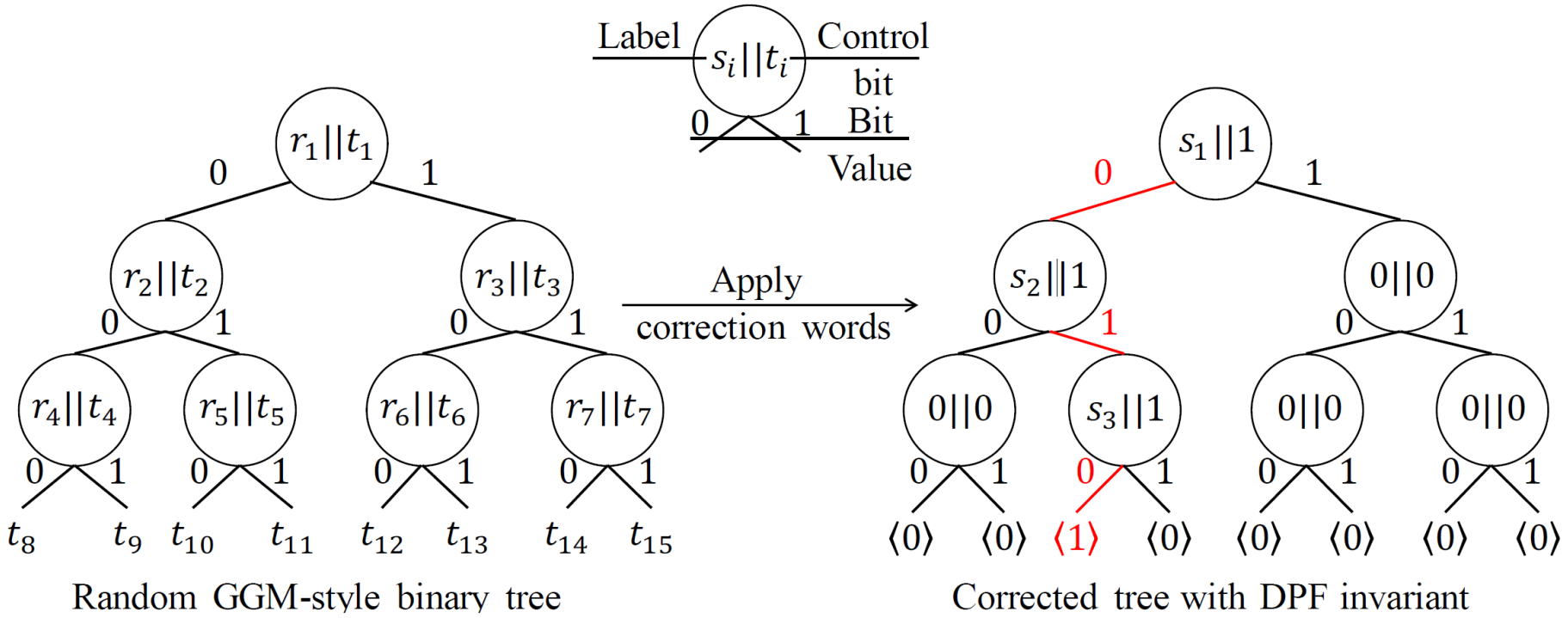
- Unable to support both arithmetic input and output.
- Potential performance bottleneck incurred by bit length.

Our Contributions

- Propose the **dealer-less FSS** scheme, including arithmetic DPF and DCF.
- FSS-based building blocks and complex function evaluation.
- Open source implementation, achieving **27~184× communication improvement** and **1.1~14× runtime improvement**



Recalling Dealer-based DPF Key Generation



Random GGM-style binary tree

Corrected tree with DPF invariant

DPF key components:

- **Root Seed**: Determine the pseudorandom GGM tree.
- **Corrections words**: Correct the tree to satisfy DPF invariant [BGI16].

Construction of Dealer-less DPF

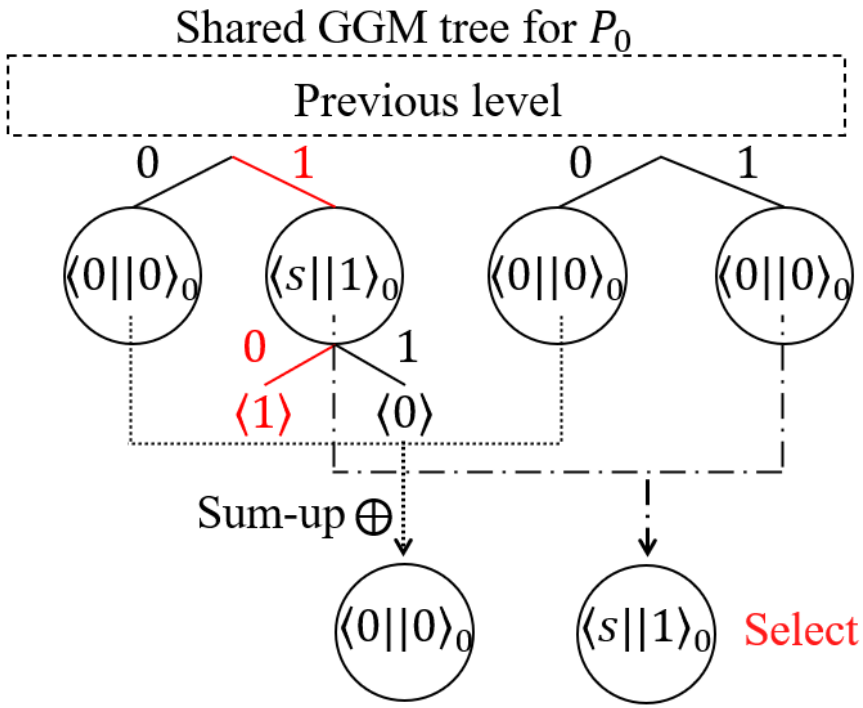
- **Traversing the GGM tree**

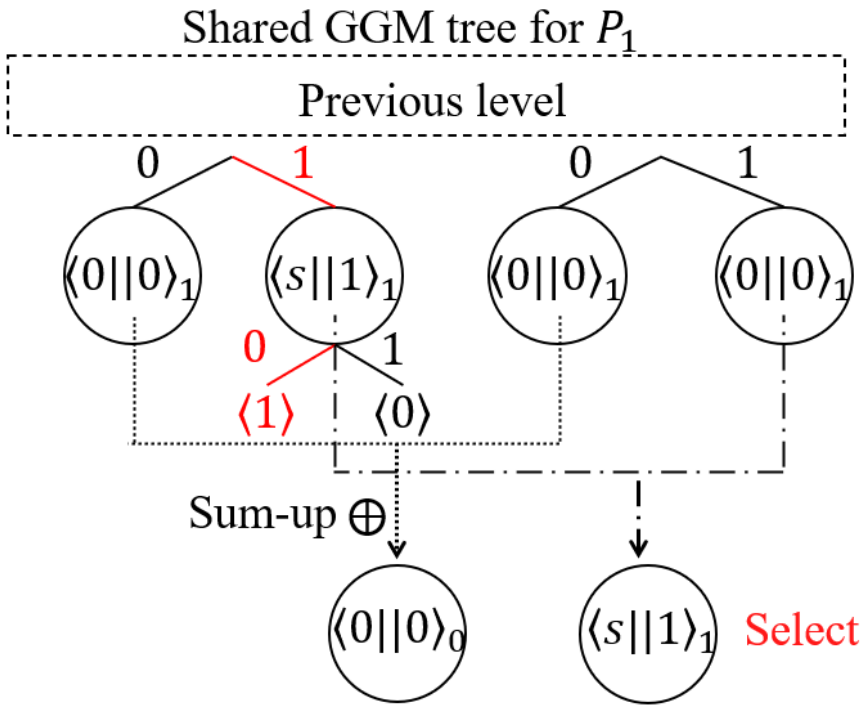
Construction of Dealer-less DPF

- **Traversing the GGM tree**
  - Problem: The *special path* is unknown to parties.

Construction of Dealer-less DPF

- **Traversing the GGM tree**
  - Problem: The *special path* is unknown to parties.
  - Solution: Sum all the left (right) nodes, off-path nodes can be cancelled.





Construction of Dealer-less DPF (cont.)

- **Supporting arithmetic output**

Construction of Dealer-less DPF (cont.)

- **Supporting arithmetic output**
  - Problem: Securely calculate the extra correction word.

$$(-1)^{t_1} \cdot (\beta - r_0 + r_1)$$

Construction of Dealer-less DPF (cont.)

- **Supporting arithmetic output**
  - Problem: Securely calculate the extra correction word.

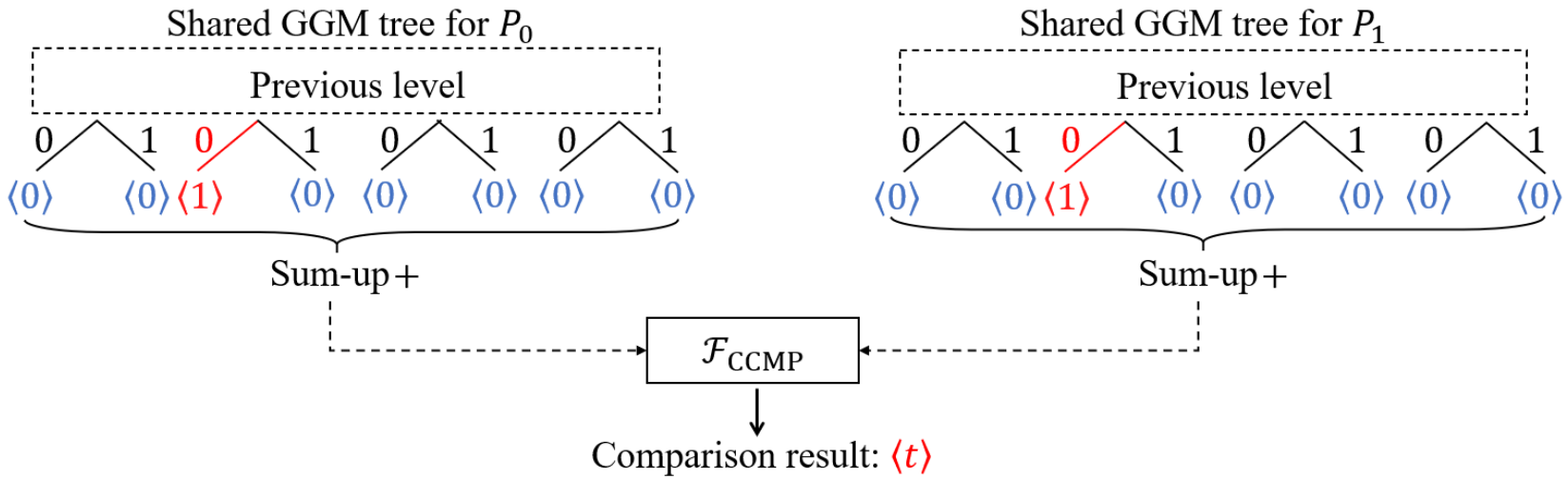
$$(-1)^{t_1} \cdot (\beta - r_0 + r_1)$$

Construction of Dealer-less DPF (cont.)

- **Supporting arithmetic output**
  - Problem: Securely calculate the extra correction word.

$$(-1)^{t_1} \cdot (\beta - r_0 + r_1)$$

  - Solution: As $t_1 = 0$ or 1, determining which party holds the larger value suffices.



Efficient Constraint Comparison

Efficient Constraint Comparison

- **Comparing two integers differs by 1**

Efficient Constraint Comparison

- **Comparing two integers differs by 1**
  - Extract the last two bits is enough.

$$x_0 = \cdots \| h_0 \| l_0$$
$$x_1 = \cdots \| h_1 \| l_1$$

Least Significant Bit

Efficient Constraint Comparison

- **Comparing two integers differs by 1**
  - Extract the last two bits is enough.

$$x_0 = \cdots \| h_0 \| l_0$$
$$x_1 = \cdots \| h_1 \| l_1$$

← Least Significant Bit

  - The comparison can be categorized as:

$$1\{x_0 < x_1\} = \begin{cases} l_1 & \text{if } h_0 = h_1 \\ \neg h_1 & \text{if } h_0 \neq h_1,\ h_0 = l_0 \\ h_1 & \text{if } h_0 \neq h_1,\ h_0 \neq l_0 \end{cases}$$

$$00\&01, 10\&11$$
$$01\&10$$
$$11\&00$$

Efficient Constraint Comparison

- **Comparing two integers differs by 1**
  - Extract the last two bits is enough.

$$x_0 = \cdots \| h_0 \| l_0$$
$$x_1 = \cdots \| h_1 \| l_1$$

← Least Significant Bit

  - The comparison can be categorized as:

$$1\{x_0 < x_1\} = \begin{cases} l_1 & \text{if } h_0 = h_1 \\ \neg h_1 & \text{if } h_0 \neq h_1,\ h_0 = l_0 \\ h_1 & \text{if } h_0 \neq h_1,\ h_0 \neq l_0 \end{cases} \quad \begin{matrix} 00\&01, 10\&11 \\ 01\&10 \\ 11\&00 \end{matrix}$$

  - Realized by **a single AND** gate:

$$l_1 \oplus (h_0 \oplus h_1) \wedge (l_1 \oplus h_1 \oplus l_0 \oplus h_0 \oplus 1)$$

Efficient Constraint Comparison

- **Comparing two integers differs by 1**
  - Extract the last two bits is enough.

$$x_0 = \cdots \| h_0 \| l_0$$
$$x_1 = \cdots \| h_1 \| l_1$$

← Least Significant Bit

  - The comparison can be categorized as:

$$1\{x_0 < x_1\} = \begin{cases} l_1 & \text{if } h_0 = h_1 \\ \neg h_1 & \text{if } h_0 \neq h_1,\ h_0 = l_0 \\ h_1 & \text{if } h_0 \neq h_1,\ h_0 \neq l_0 \end{cases} \quad \begin{matrix} 00\&01, 10\&11 \\ 01\&10 \\ 11\&00 \end{matrix}$$

  - Realized by **a single AND** gate:

$$l_1 \oplus (h_0 \oplus h_1) \wedge (l_1 \oplus h_1 \oplus l_0 \oplus h_0 \oplus 1)$$

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_{α}:

$$V_{CW} \coloneqq (-1)^{t_1^{(i-1)}}(C_{\mathbb{G}}(v_1^{\text{Lose}}) - C_{\mathbb{G}}(v_0^{\text{Lose}}) - V_{\alpha} + \alpha[i] \cdot \beta)$$

$$V_{\alpha} \coloneqq C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}}) - C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}}) + \alpha[i] \cdot \beta)$$

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} \left(\underbrace{C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right) - C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right)}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}}\right)$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Keep}}\right) - C_{\mathbb{G}}\left(v_1^{\text{Keep}}\right)}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right)}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta}_{(\mathcal{F}_{\text{MUL}})}$$

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} \left(\underbrace{C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right) - C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right)}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}}\right)$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Keep}}\right) - C_{\mathbb{G}}\left(v_1^{\text{Keep}}\right)}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right)}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta}_{(\mathcal{F}_{\text{MUL}})}$$

  - Optimization: Reduced multiplexer.

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} (\underbrace{C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right) - C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right)}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}})$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Keep}}\right) - C_{\mathbb{G}}\left(v_1^{\text{Keep}}\right)}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right)}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta)}_{(\mathcal{F}_{\text{MUL}})}$$

  - Optimization: Reduced multiplexer.
    - Local Compute: $-C_{\mathbb{G}}\left(v_0^{\text{Left}}\right) - C_{\mathbb{G}}\left(v_0^{\text{Right}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Left}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Right}}\right)$

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} (\underbrace{C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right) - C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right)}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}})$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Keep}}\right) - C_{\mathbb{G}}\left(v_1^{\text{Keep}}\right)}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}\left(v_0^{\text{Lose}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Lose}}\right)}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta)}_{(\mathcal{F}_{\text{MUL}})}$$

  - Optimization: Reduced multiplexer.
    - Local Compute: $-C_{\mathbb{G}}\left(v_0^{\text{Left}}\right) - C_{\mathbb{G}}\left(v_0^{\text{Right}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Left}}\right) + C_{\mathbb{G}}\left(v_1^{\text{Right}}\right)$
    - As *Keep=Left* or *Keep=Right*, the above equals:

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} (\underbrace{C_{\mathbb{G}}(v_1^{\text{Lose}}) - C_{\mathbb{G}}(v_0^{\text{Lose}})}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}})$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}})}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}})}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta}_{(\mathcal{F}_{\text{MUL}})})$$

  - Optimization: Reduced multiplexer.
    - Local Compute: $-C_{\mathbb{G}}(v_0^{\text{Left}}) - C_{\mathbb{G}}\left(v_0^{\text{Right}}\right) + C_{\mathbb{G}}(v_1^{\text{Left}}) + C_{\mathbb{G}}\left(v_1^{\text{Right}}\right)$
    - As *Keep=Left* or *Keep=Right*, the above equals:

$$C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}}) - C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}})$$

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} (\underbrace{C_{\mathbb{G}}(v_1^{\text{Lose}}) - C_{\mathbb{G}}(v_0^{\text{Lose}})}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}})$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}})}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}})}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta}_{(\mathcal{F}_{\text{MUL}})})$$

  - Optimization: Reduced multiplexer.
    - Local Compute: $-C_{\mathbb{G}}(v_0^{\text{Left}}) - C_{\mathbb{G}}\left(v_0^{\text{Right}}\right) + C_{\mathbb{G}}(v_1^{\text{Left}}) + C_{\mathbb{G}}\left(v_1^{\text{Right}}\right)$
    - As *Keep=Left* or *Keep=Right*, the above equals:

$$\underbrace{C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}})}_{} - \underbrace{C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}})}_{}$$

Construction of Dealer-less DCF

- **Generating extra comparison correction words [BCG21] via 2PC**
  - Computing V_{CW} and V_α:

$$V_{CW} \coloneqq \underbrace{(-1)^{t_1^{(i-1)}}}_{\mathcal{F}_{\text{CCMP}}} (\underbrace{C_{\mathbb{G}}(v_1^{\text{Lose}}) - C_{\mathbb{G}}(v_0^{\text{Lose}})}_{\mathcal{F}_{\text{MUX}}} - V_\alpha + \underbrace{\alpha[i] \cdot \beta}_{\mathcal{F}_{\text{MUL}}})$$

$$V_\alpha \coloneqq \underbrace{C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}})}_{\mathcal{F}_{\text{MUX}}} - \underbrace{C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}})}_{(\mathcal{F}_{\text{MUX}})} + \underbrace{\alpha[i] \cdot \beta)}_{(\mathcal{F}_{\text{MUL}})}$$

  - Optimization: Reduced multiplexer.
    - Local Compute: $-C_{\mathbb{G}}(v_0^{\text{Left}}) - C_{\mathbb{G}}\left(v_0^{\text{Right}}\right) + C_{\mathbb{G}}(v_1^{\text{Left}}) + C_{\mathbb{G}}\left(v_1^{\text{Right}}\right)$
    - As *Keep=Left* or *Keep=Right*, the above equals:

$$C_{\mathbb{G}}(v_0^{\text{Keep}}) - C_{\mathbb{G}}(v_1^{\text{Keep}}) - C_{\mathbb{G}}(v_0^{\text{Lose}}) + C_{\mathbb{G}}(v_1^{\text{Lose}})$$

    - Just locally subtract previous multiplexer output to get what we need now.

Basic Building Blocks

Basic Building Blocks

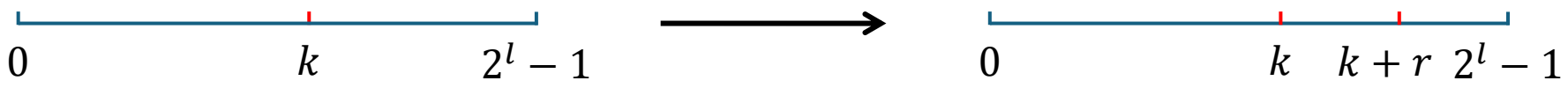
- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$

Basic Building Blocks

- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$

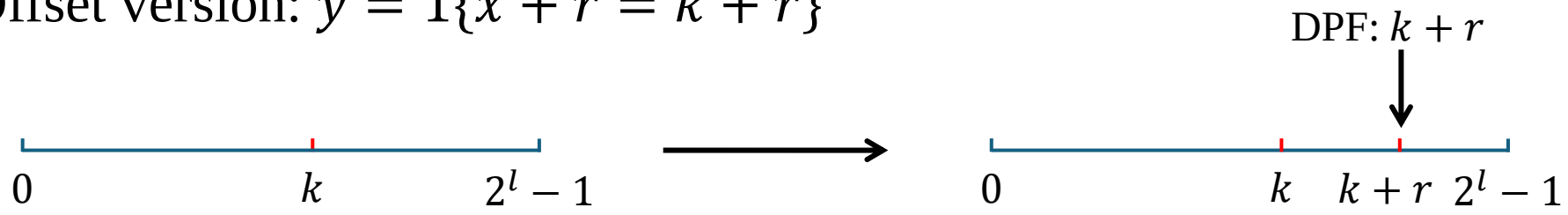
Basic Building Blocks

- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



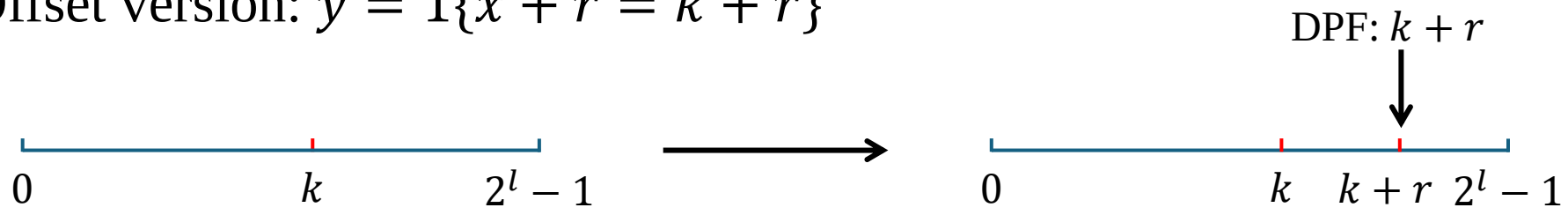
Basic Building Blocks

- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



Basic Building Blocks

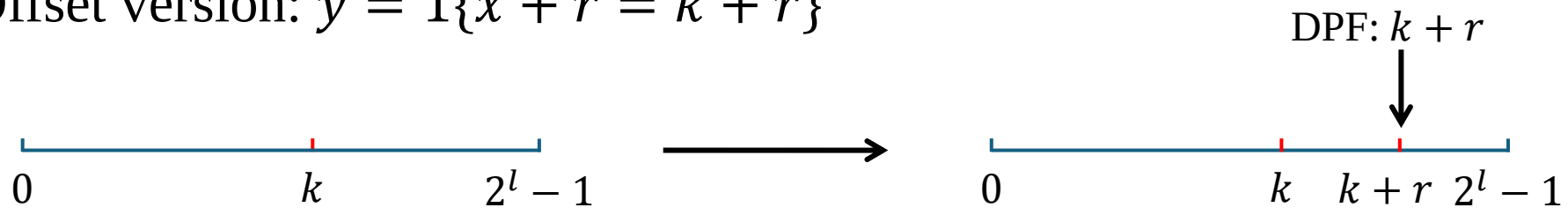
- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



- **Comparison:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x < k\}$

Basic Building Blocks

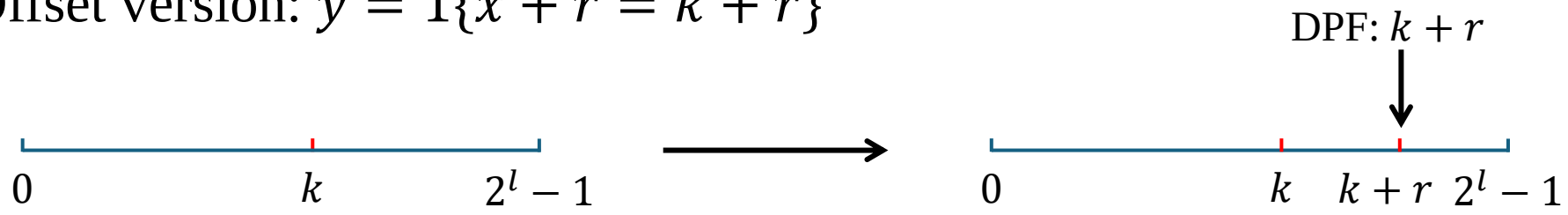
- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



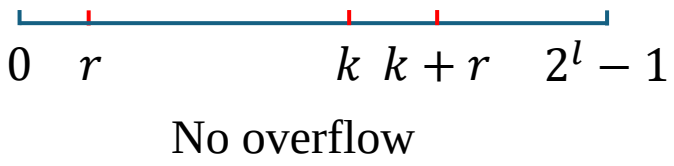
- **Comparison:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x < k\}$
  - Problem: an overflowed $k + r$ lead to incorrect result!

Basic Building Blocks

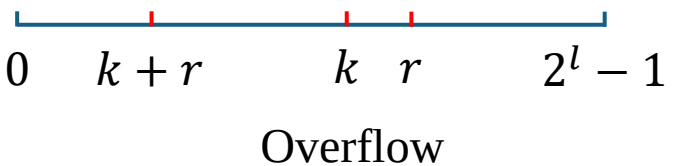
- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



- **Comparison:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x < k\}$
  - Problem: an overflowed $k + r$ lead to incorrect result!



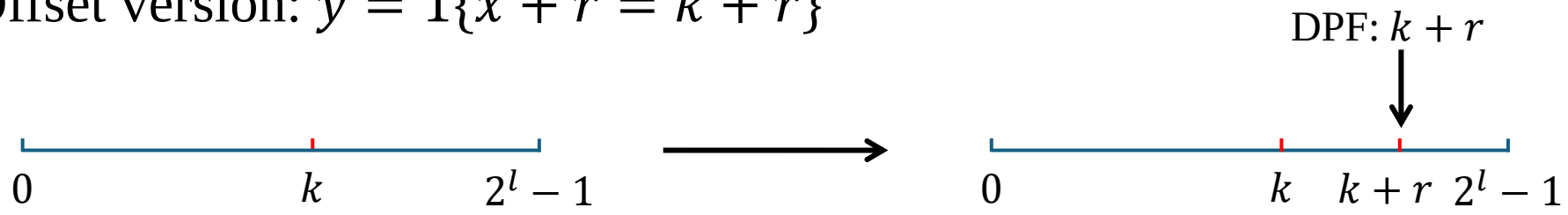
No overflow



Overflow

Basic Building Blocks

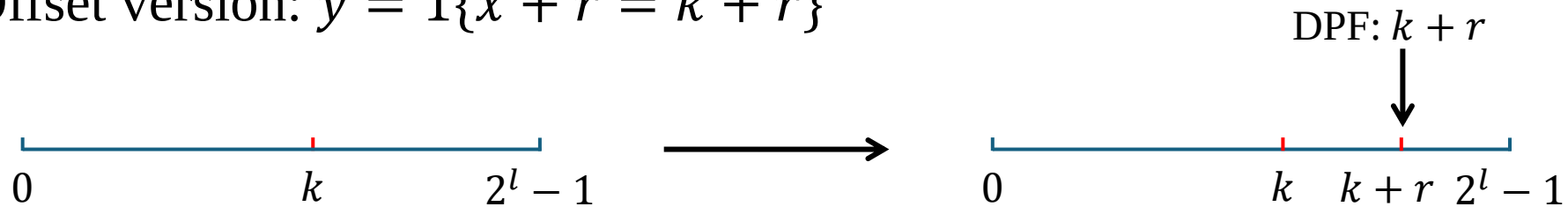
- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



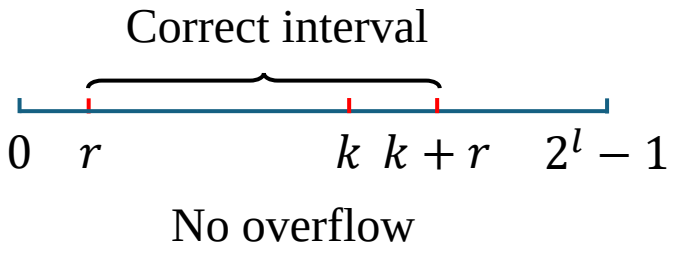
- **Comparison:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x < k\}$
  - Problem: an overflowed $k + r$ lead to incorrect result!

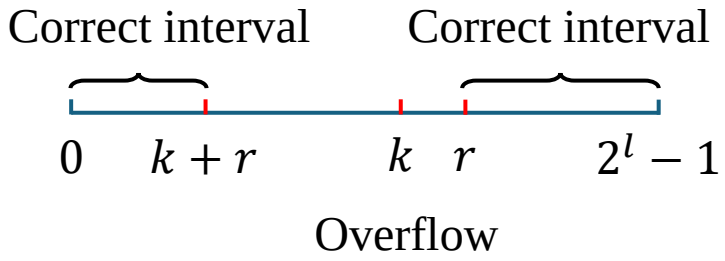
Basic Building Blocks

- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



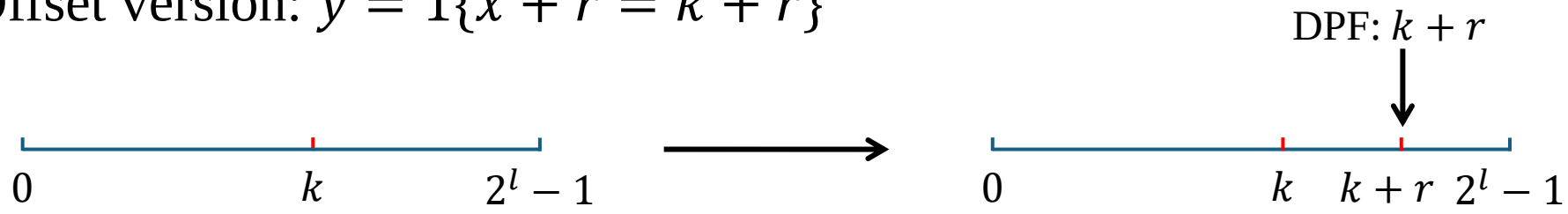
- **Comparison:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x < k\}$
  - Problem: an overflowed $k + r$ lead to incorrect result!





Basic Building Blocks

- **Equality test:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x = k\}$
  - Offset version: $y = 1\{x + r = k + r\}$



- **Comparison:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = 1\{x < k\}$
  - Problem: an overflowed $k + r$ lead to incorrect result!

  - Solution: Check if $k > k + r$ at offline stage.

Building Blocks

Building Blocks

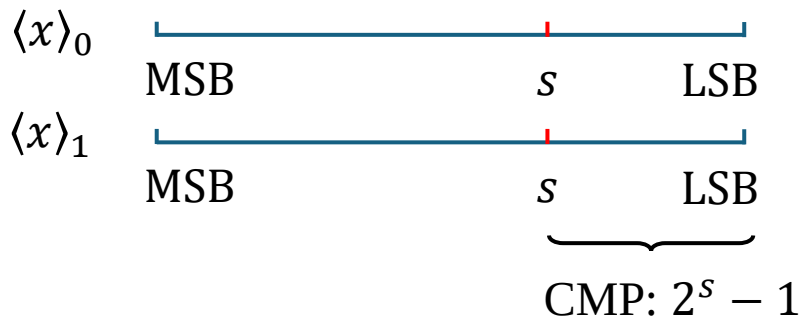
- **Truncation-and-reduce:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = x \gg s \in \mathbb{Z}_{2^{l-s}}$

Building Blocks

- **Truncation-and-reduce:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = x \gg s \in \mathbb{Z}_{2^{l-s}}$
  - Drop the lower bit, check potential overflow.

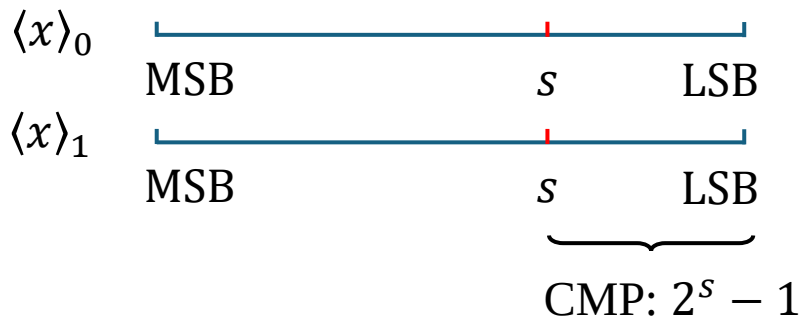
Building Blocks

- **Truncation-and-reduce:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = x \gg s \in \mathbb{Z}_{2^{l-s}}$
  - Drop the lower bit, check potential overflow.



Building Blocks

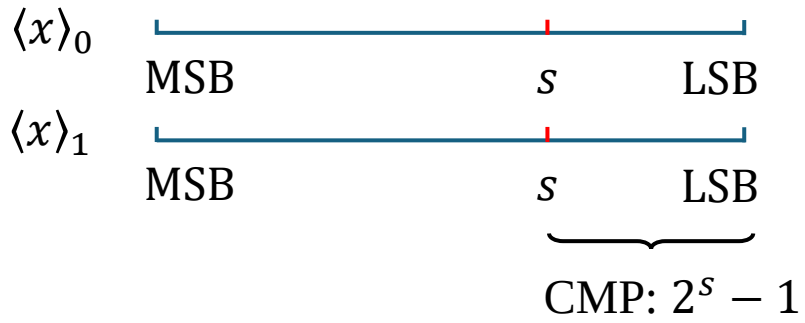
- **Truncation-and-reduce:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = x \gg s \in \mathbb{Z}_{2^{l-s}}$
  - Drop the lower bit, check potential overflow.



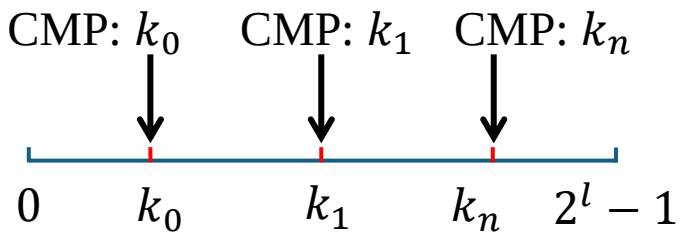
- **Interval containment**: given $\langle x \rangle$, output $\overrightarrow{\langle y \rangle}$ where $y[i] = 1\{x \in I_i\}$

Building Blocks

- **Truncation-and-reduce:** given $\langle x \rangle$, output $\langle y \rangle$ where $y = x \gg s \in \mathbb{Z}_{2^{l-s}}$
  - Drop the lower bit, check potential overflow.



- **Interval containment**: given $\langle x \rangle$, output $\overrightarrow{\langle y \rangle}$ where $y[i] = 1\{x \in I_i\}$
  - For each interval, check compare with the endpoints.



Building Blocks (cont.)

- **Digit Decomposition:** given $\langle x \rangle$, output $\{x_{k-1}, \ldots, x_0\}$ satisfying $x = x_{k-1} || \ldots || x_0$

Building Blocks (cont.)

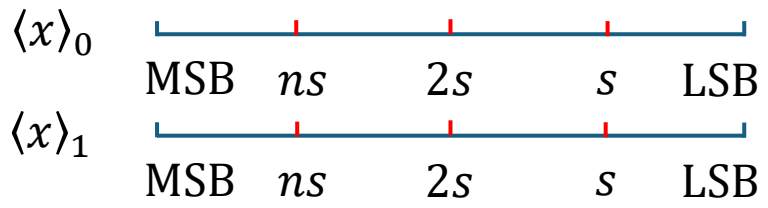
- **Digit Decomposition:** given $\langle x \rangle$, output $\{x_{k-1}, \ldots, x_0\}$ satisfying $x = x_{k-1} || \ldots || x_0$
  - Local decompose, check potential overflow.

Building Blocks (cont.)

- **Digit Decomposition:** given $\langle x \rangle$, output $\{x_{k-1}, \ldots, x_0\}$ satisfying $x = x_{k-1} || \ldots || x_0$
  - Local decompose, check potential overflow.
  - Overflow incurred by:
    - Current segment overflow.
    - Carryover from preceding segment.

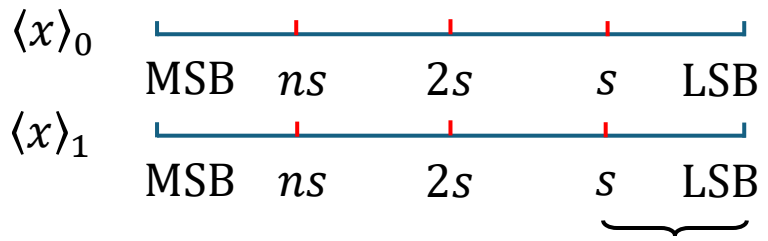
Building Blocks (cont.)

- **Digit Decomposition:** given $\langle x \rangle$, output $\{x_{k-1}, \ldots, x_0\}$ satisfying $x = x_{k-1} || \ldots || x_0$
  - Local decompose, check potential overflow.
  - Overflow incurred by:
    - Current segment overflow.
    - Carryover from preceding segment.



Building Blocks (cont.)

- **Digit Decomposition:** given $\langle x \rangle$, output $\{x_{k-1}, \dots, x_0\}$ satisfying $x = x_{k-1} || \dots || x_0$
  - Local decompose, check potential overflow.
  - Overflow incurred by:
    - Current segment overflow.
    - Carryover from preceding segment.



For each segment: CMP: $2^s - 1$, EQ: $2^s - 1$

Applications in Scientific Computing

Applications in Scientific Computing

- **Lookup Table**

Applications in Scientific Computing

- **Lookup Table**
  - Public table: Invoking equality test for every position.
  - Private table: Generate equality test key with payload being the table entry.

Applications in Scientific Computing

- **Lookup Table**
  - Public table: Invoking equality test for every position.
  - Private table: Generate equality test key with payload being the table entry.

- **Spline polynomial approximation**

Applications in Scientific Computing

- **Lookup Table**
  - Public table: Invoking equality test for every position.
  - Private table: Generate equality test key with payload being the table entry.

- **Spline polynomial approximation**
  - Generating coefficients from $\sum a_i \cdot x^i$ to $\sum a_i \cdot (x - r)^i$ via secure multiplication.

Applications in Scientific Computing

- **Lookup Table**
  - Public table: Invoking equality test for every position.
  - Private table: Generate equality test key with payload being the table entry.

- **Spline polynomial approximation**
  - Generating coefficients from $\sum a_i \cdot x^i$ to $\sum a_i \cdot (x - r)^i$ via secure multiplication.
  - Fetching the coefficient for the spline.

Applications in Scientific Computing

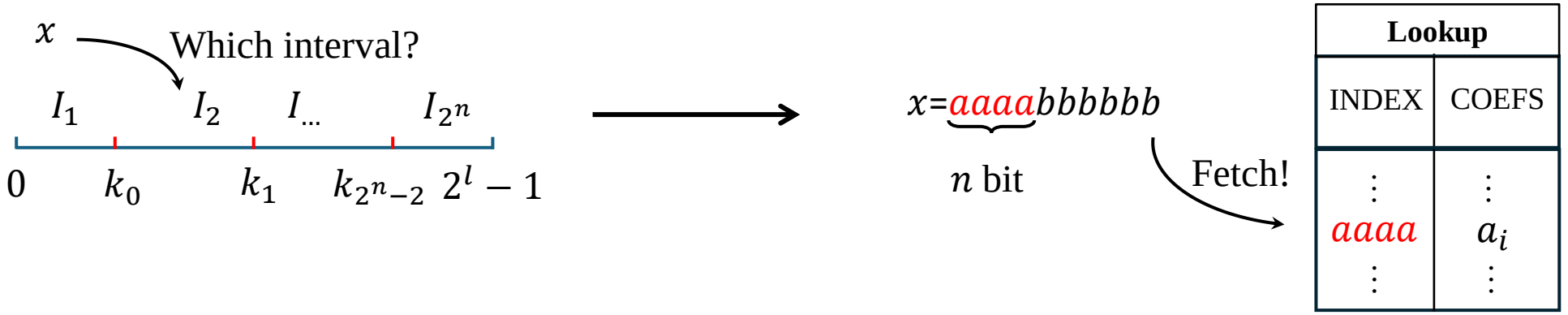
- **Lookup Table**
  - Public table: Invoking equality test for every position.
  - Private table: Generate equality test key with payload being the table entry.

- **Spline polynomial approximation**
  - Generating coefficients from $\sum a_i \cdot x^i$ to $\sum a_i \cdot (x - r)^i$ via secure multiplication.
  - Fetching the coefficient for the spline.
    - Optimization: We do not need costly interval containment here.

Applications in Scientific Computing

- **Lookup Table**
  - Public table: Invoking equality test for every position.
  - Private table: Generate equality test key with payload being the table entry.

- **Spline polynomial approximation**
  - Generating coefficients from $\sum a_i \cdot x^i$ to $\sum a_i \cdot (x-r)^i$ via secure multiplication.
  - Fetching the coefficient for the spline.
    - Optimization: We do not need costly interval containment here.

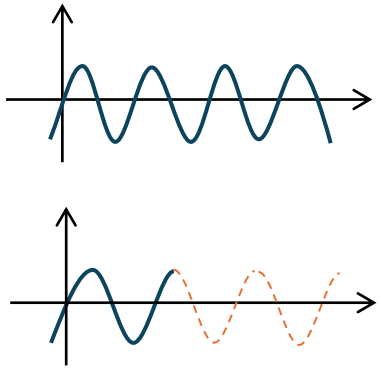


Applications in Scientific Computing (cont.)

- **Trigonometric evaluation**

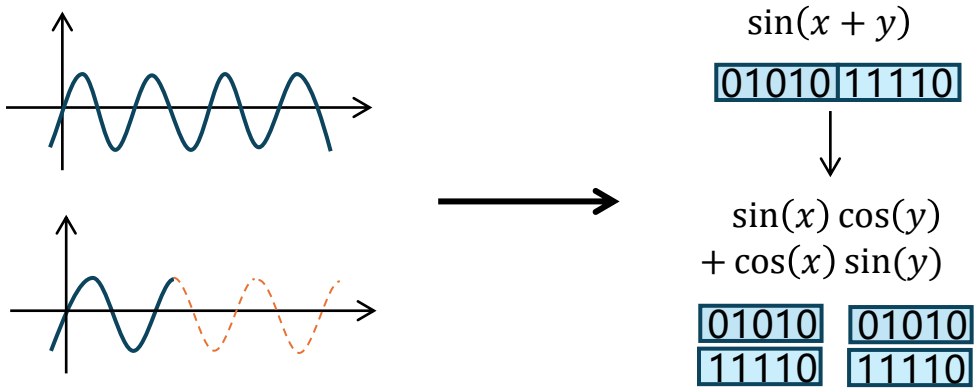
Applications in Scientific Computing (cont.)

- **Trigonometric evaluation**
  - Observation-1: Leveraging periodic properties to reduce bit length.
    - Range reduction: From *full domain* to *one period*.
    - Periodic Reflection: From *one period* to *¼ period*.



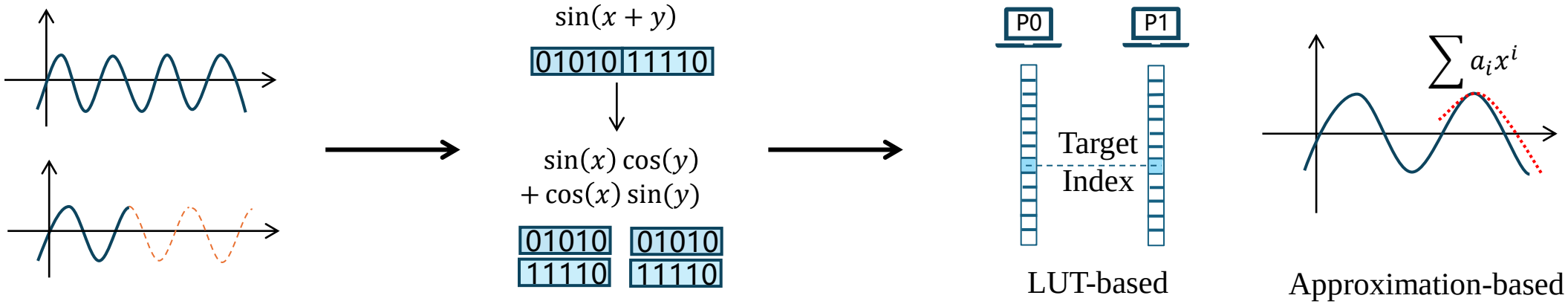
Applications in Scientific Computing (cont.)

- **Trigonometric evaluation**
  - Observation-1: Leveraging periodic properties to reduce bit length.
    - Range reduction: From *full domain* to *one period*.
    - Periodic Reflection: From *one period* to *¼ period*.
  - Observation-2: Leveraging *sum-to-product* identity.
    - Specialized Transformation: Digit decomposition applicable.



Applications in Scientific Computing (cont.)

- **Trigonometric evaluation**
  - Observation-1: Leveraging periodic properties to reduce bit length.
    - Range reduction: From *full domain* to *one period*.
    - Periodic Reflection: From *one period* to *¼ period*.
  - Observation-2: Leveraging *sum-to-product* identity.
    - Specialized Transformation: Digit decomposition applicable.



Evaluation

- **Experimental results of dealer-less DPF and DCF**

Protocol	Params.	Time LAN		Time WAN		Comm.	
	ℓ	Gen(s)	Eval(μs)	Gen(s)	Eval(μs)	Gen(MB)	Eval(B)
Π_{DPF}	8	0.050	0.577	1.288	0.615	0.046	0
	16	0.092	1.150	2.450	1.151	0.080	0
	18	0.124	1.278	2.780	1.265	0.088	0
Π_{DCF}	8	0.085	1.818	2.417	1.866	0.145	0
	16	0.180	3.357	4.625	3.505	0.277	0
	18	0.280	3.739	5.247	3.718	0.310	0

Evaluation

- **Experimental results of trigonometric evaluation**

Protocol	Impl.	Runtime		Comm.	Error
		LAN (ms)	WAN (s)	(KB)	(ULP)
Π_{Sin}	Ours (LUT)	0.612	0.164	0.050	1.477
	Ours (Approx)	**0.329**	**0.133**	**0.035**	1.387
	MP-SPDZ	0.941	0.753	1.560	0.629
	EzPC-Secfloat	1.614	0.380	26.014	**0.318**
Π_{Cos}	Ours (LUT)	0.612	0.164	0.050	0.360
	Ours (Approx)	**0.329**	**0.133**	**0.035**	1.117
	MP-SPDZ	0.892	0.713	1.528	1.070
	EzPC-Secfloat	1.625	0.382	26.080	**0.318**
Π_{Tan}	Ours (LUT)	**0.307**	**0.092**	**0.021**	**0**
	Ours (Approx)	0.309	0.133	0.025	1.053
	MP-SPDZ	1.761	1.357	3.840	5.088
	EzPC-Secfloat	2.185	0.533	36.043	0.244

Evaluation

- **Experimental results of case studies on *proximity test* and *biometric authentication***

Protocol	Impl.	Runtime		Comm.	Error
		LAN (ms)	WAN (s)	(KB)	(ULP)
Biometric Authentication	Ours (LUT)	1.429	**0.369**	**0.084**	**0**
	Ours (Approx)	**1.252**	0.530	0.100	2.549
	MP-SPDZ	5.977	5.339	15.360	3.707
	EzPC-Secfloat	9.369	2.372	165.293	2.370
Proximity Test	Ours (LUT)	2.409	0.697	0.220	2.094
	Ours (Approx)	**1.394**	**0.572**	**0.160**	**0.926**
	MP-SPDZ	1.977	1.694	9.424	2.266
	EzPC-Secfloat	8.210	2.030	173.322	6.299

Thank You

Pengzhi Xing, *p.xing@std.uestc.edu.cn*