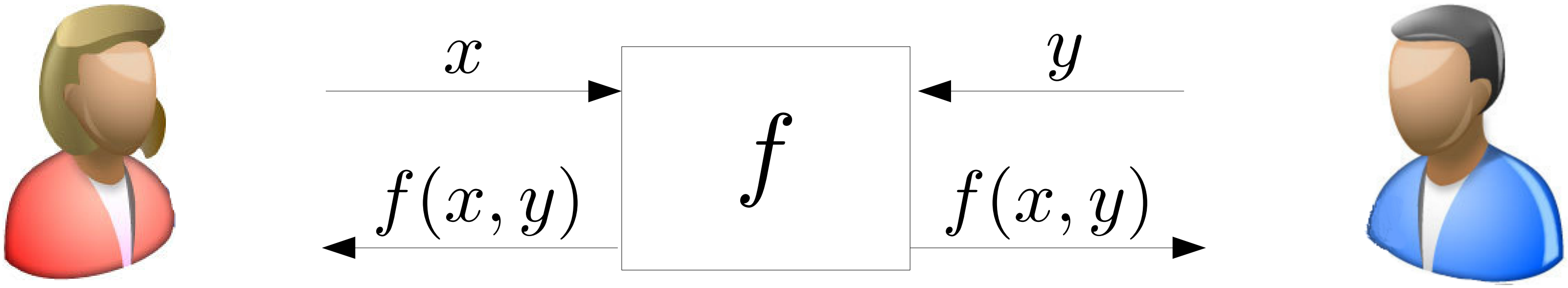
Pushing the Communication Barrier in 2PC using Lookup Tables

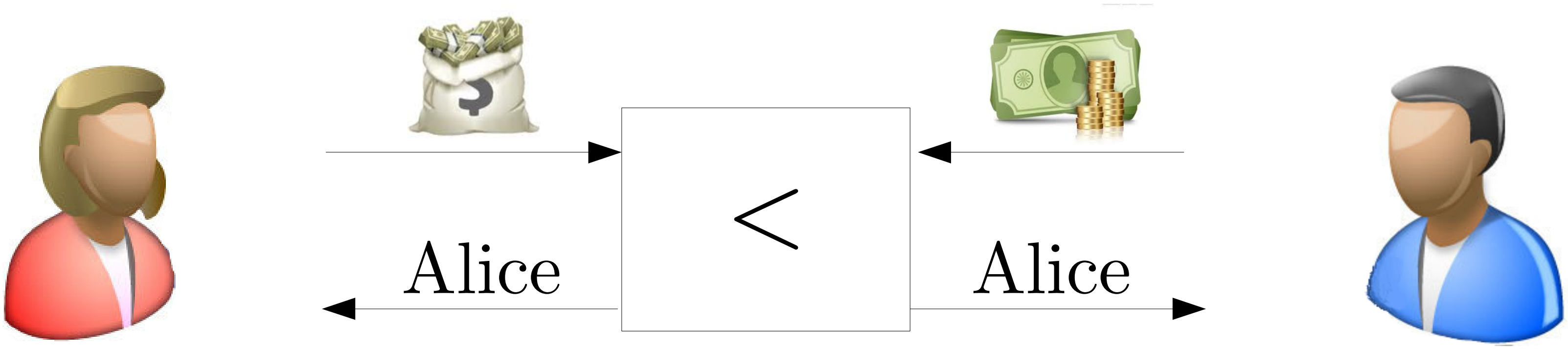
Ghada Dessouky*, Farinaz Koushanfar†, Ahmad-Reza Sadeghi*,
Thomas Schneider*, Shaza Zeitouni*, and Michael Zohner*

*Technische Universität Darmstadt
†University of California, San Diego

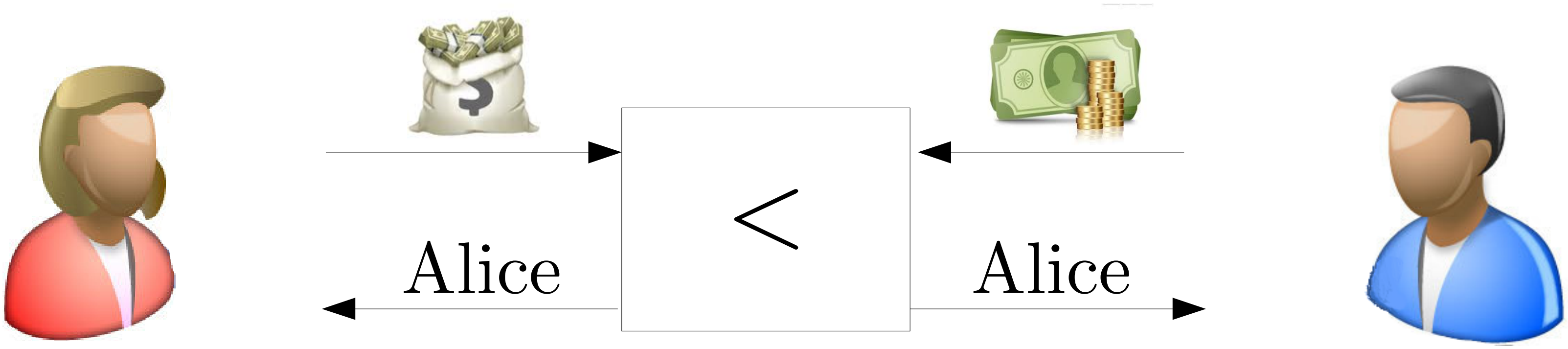
Secure 2PC



Secure 2PC



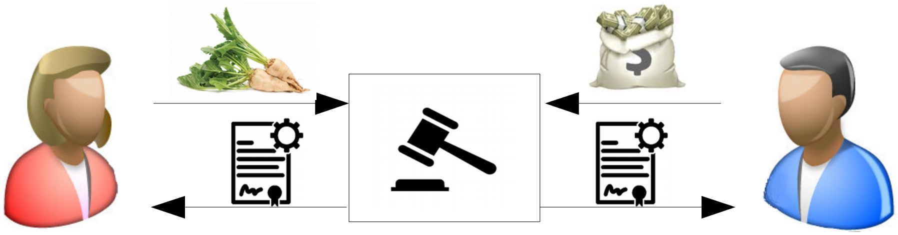
Secure 2PC



This work: semi-honest (passive) security

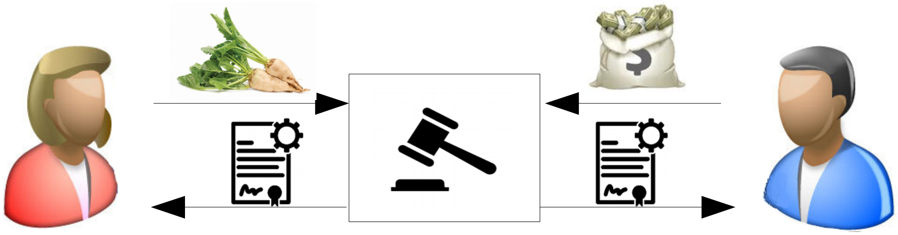
Applications of Secure 2PC

Sugar Beet Auction [BCD+09]

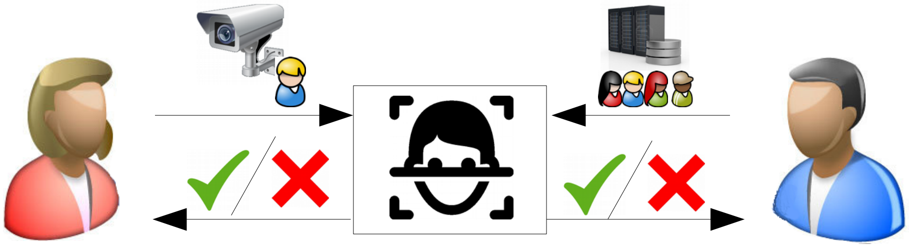


Applications of Secure 2PC

Sugar Beet Auction [BCD+09]

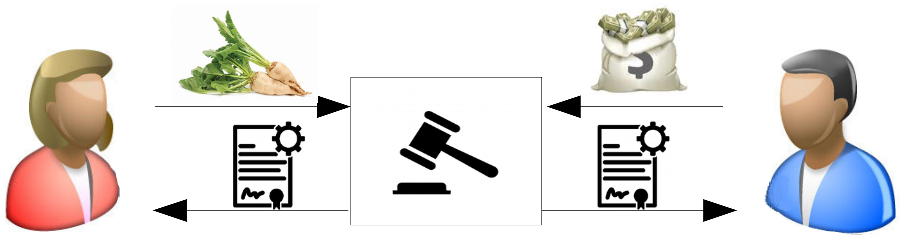


Face Recognition [EFG+09]

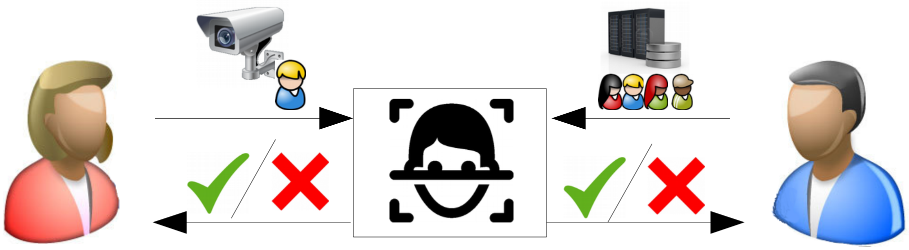


Applications of Secure 2PC

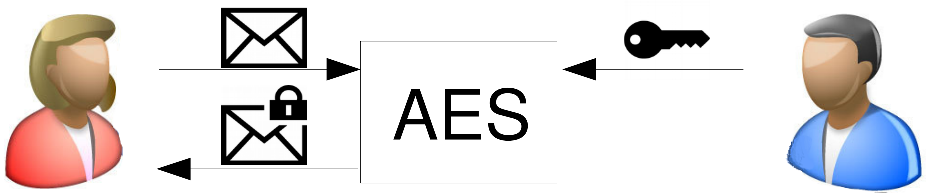
Sugar Beet Auction [BCD+09]



Face Recognition [EFG+09]

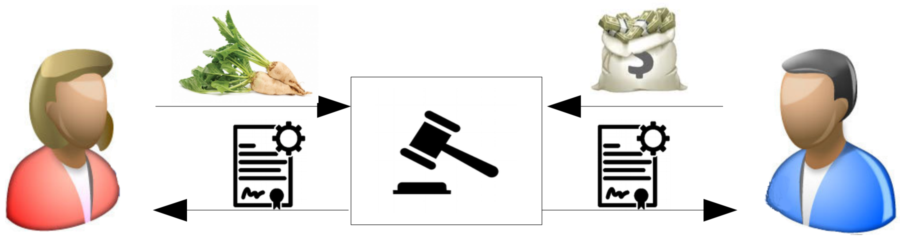


Blind En/Decryption [Dyadic]

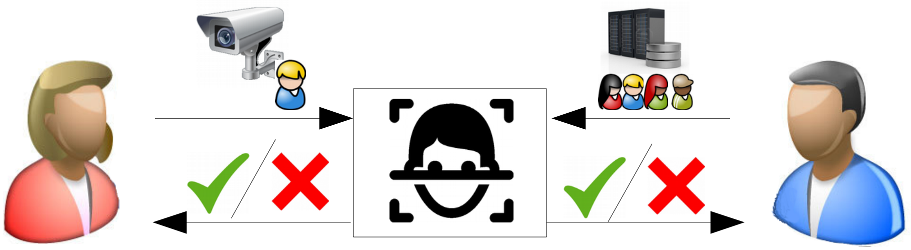


Applications of Secure 2PC

Sugar Beet Auction [BCD+09]

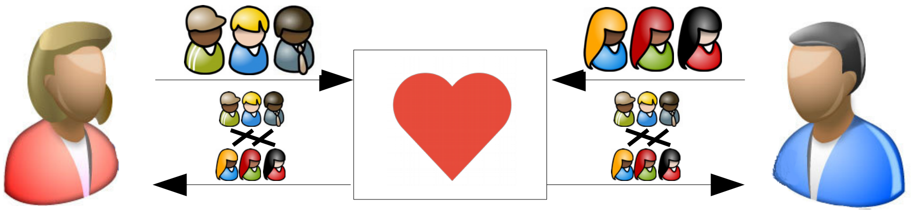


Face Recognition [EFG+09]



Blind En/Decryption [Dyadic]

Stable Matching [DES16]



Generic Secure 2PC

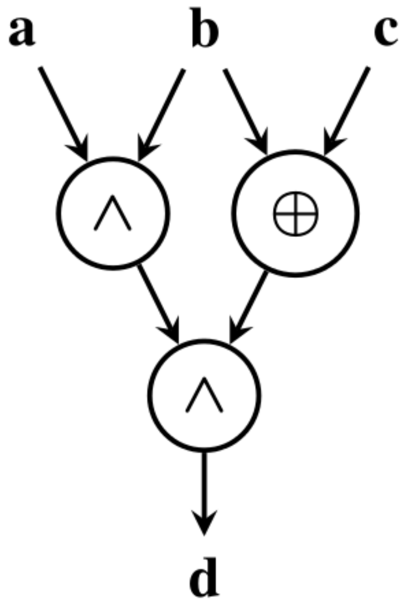
Two prominent techniques: Yao's protocol and GMW

Both evaluate Boolean circuits securely

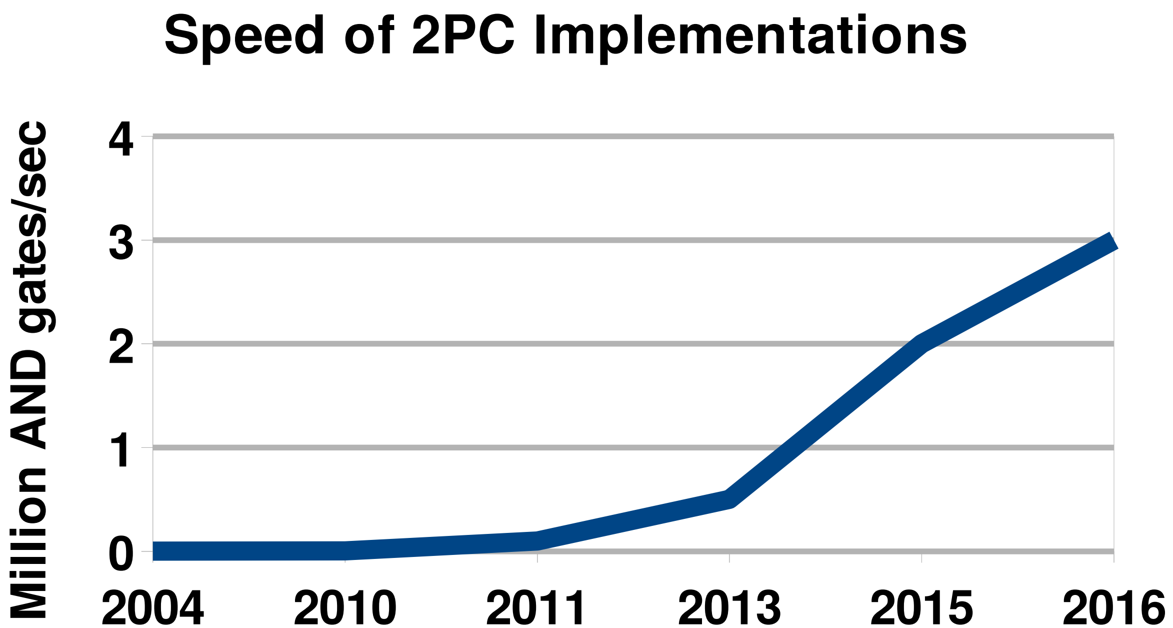
- XOR gates are „free“
- AND gates cost sym. crypto / comm.

Difference: round complexity

- Yao is constant round
- GMW requires interaction per AND gate



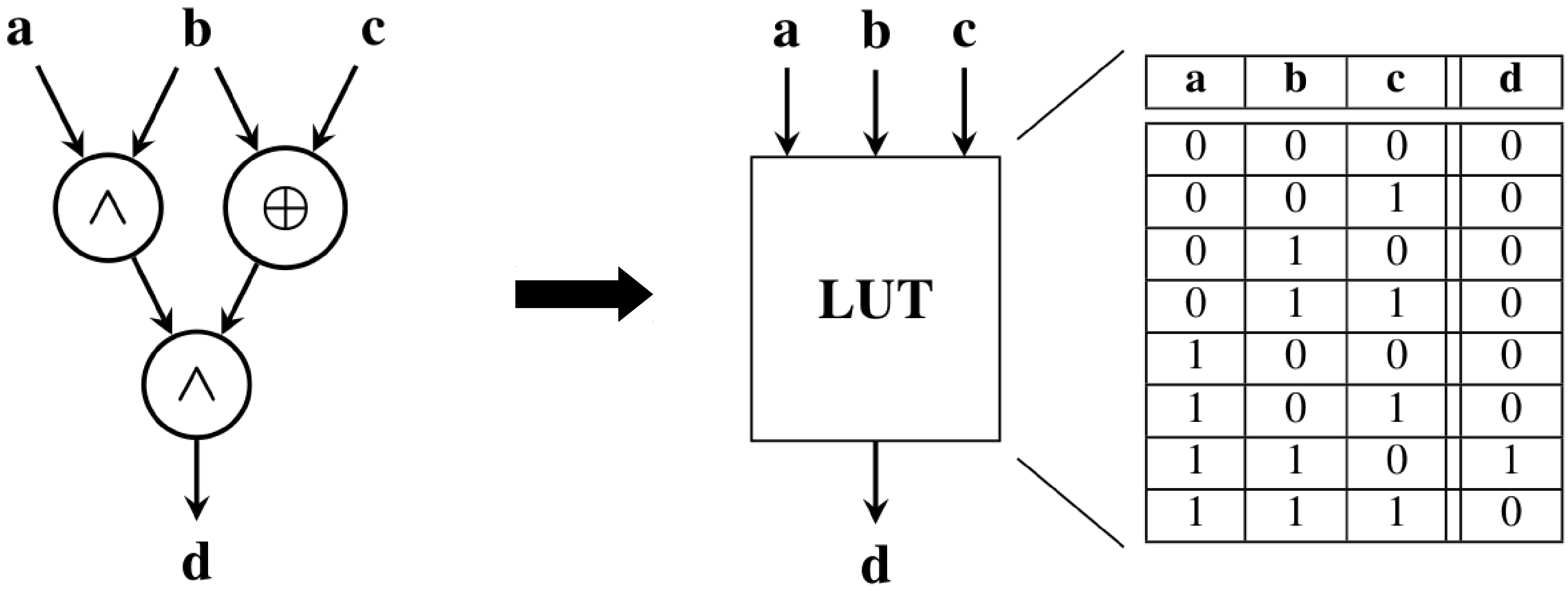
Practical Improvements



Currently: 3 million ANDs/s per thread, however:

- We have hit a comm. lower-bound per AND for Yao [ZRE15]
- Run-time for GMW often is mostly network latency

Lookup Tables



a	b	c	d
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Our Contributions

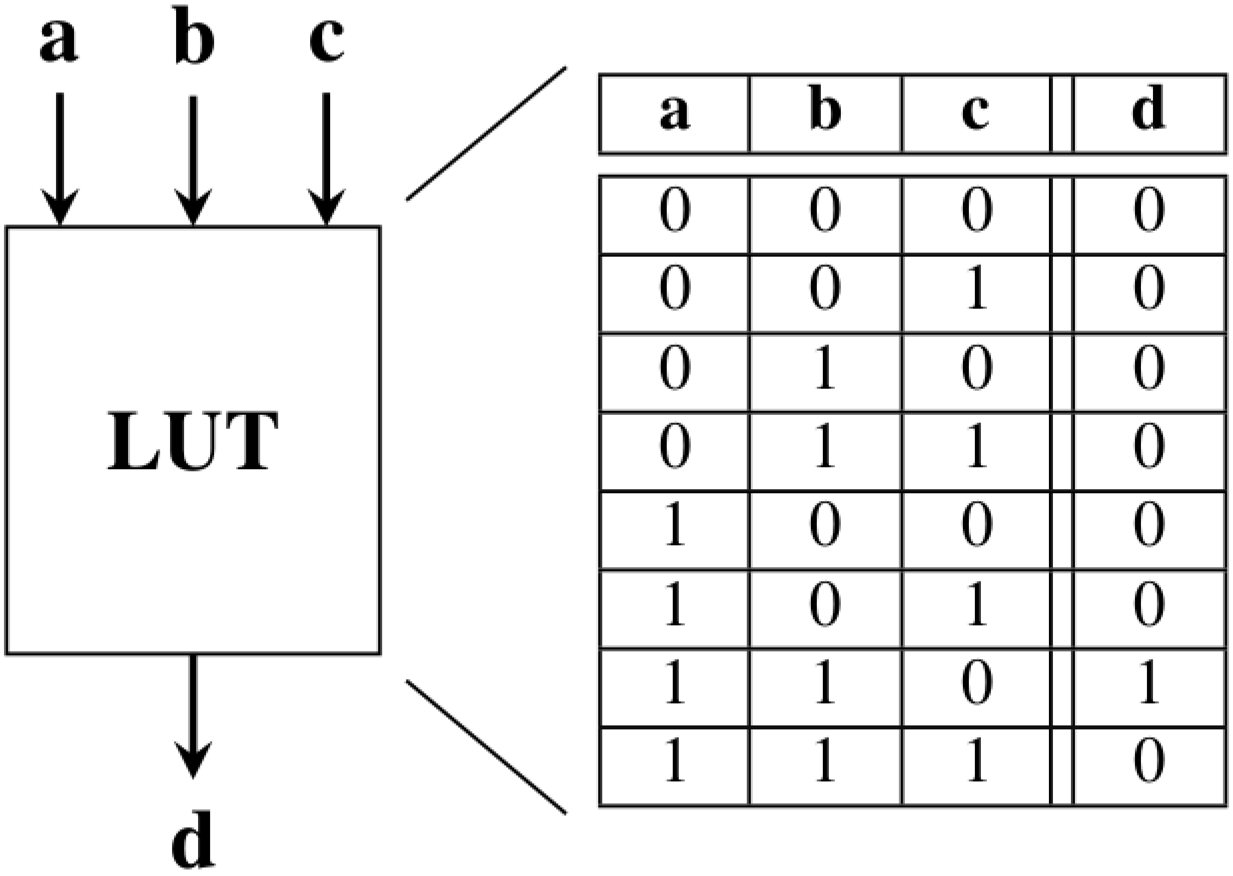
Develop lookup table (LUT)-based protocols

Tool support for generating LUT circuits

Evaluation and comparison

(Paper: improve building blocks & comm. for GMW)

Lookup Table Protocols

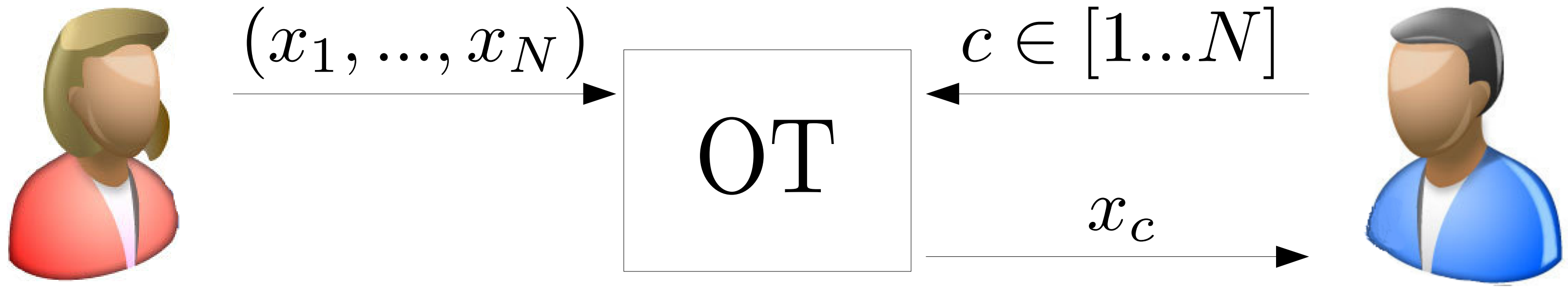


a	b	c	d
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

1ooN Oblivious Transfer

Bob obliviously obtains one of N messages s.t.

- Alice does not learn Bob's choice c
- Bob does not learn Alice's other messages



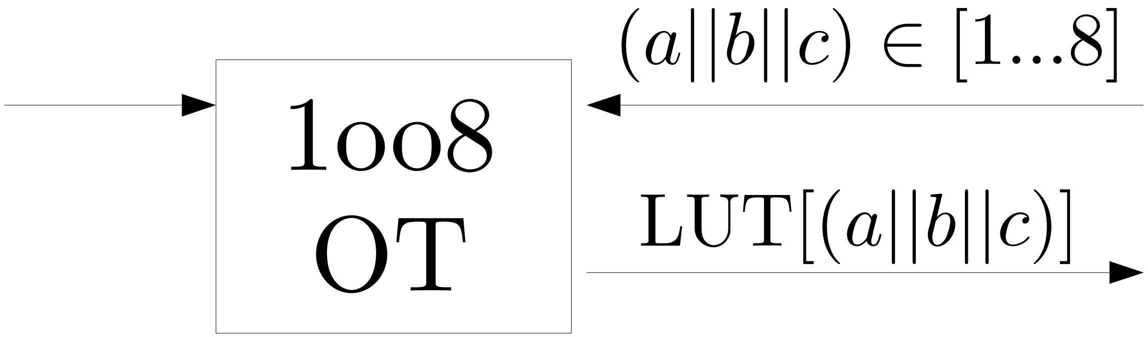
Most efficient protocol 1ooN OT: [KK13]

Intuition of the Protocols

Use [KK13] 1ooN OT to perform table lookups

LUT:

a	b	c	d
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



1oo8
OT

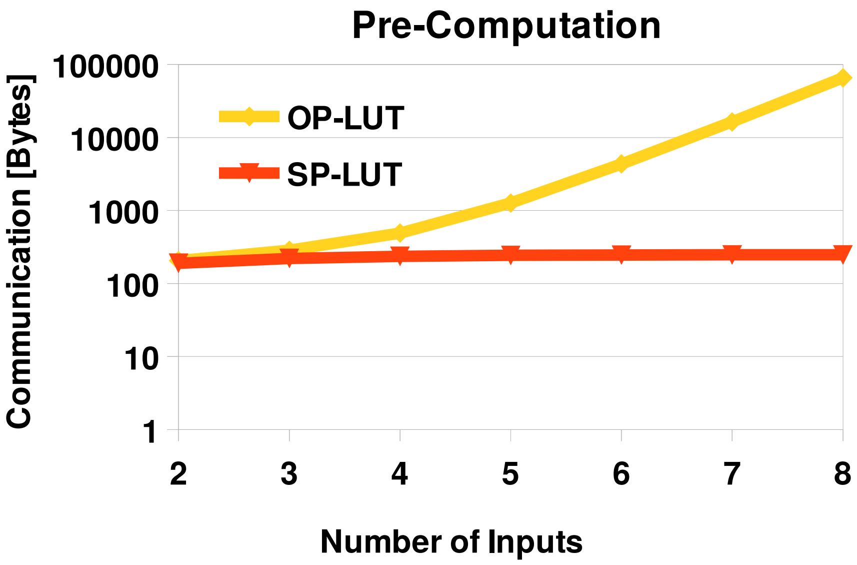
$(a||b||c) \in [1...8]$

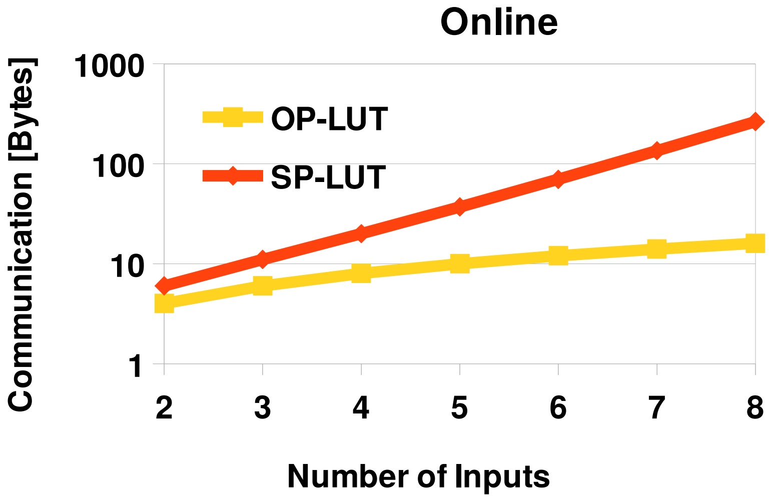
$\mathrm{LUT}[(a||b||c)]$

LUT Protocols

We develop two LUT protocols based on [KK13] OT

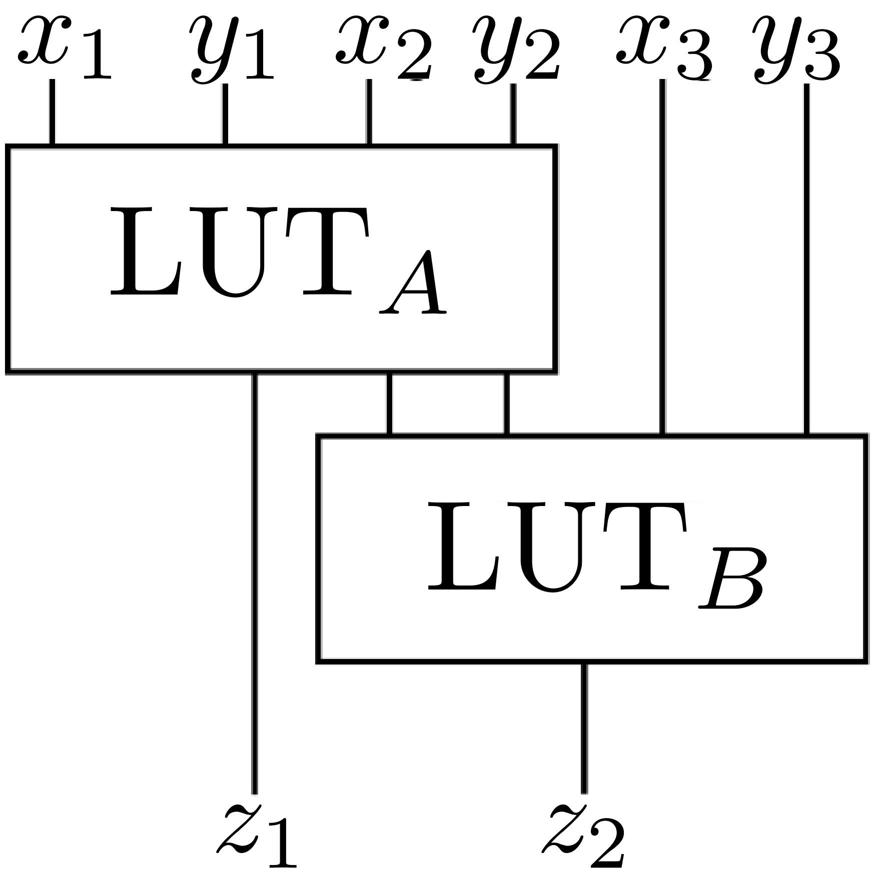
- Online Phase LUT (OP-LUT)
- Setup Phase LUT (SP-LUT)





Generating LUT Circuits

$$z = x + y$$



Tool Support for LUTs

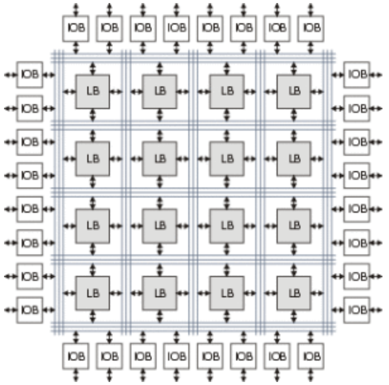
Generating LUT circuits is difficult and error-prone

- Automation is required

Idea: FPGAs internally operate on single output LUTs

- Use ABC logic syntesis to generate single output LUTs

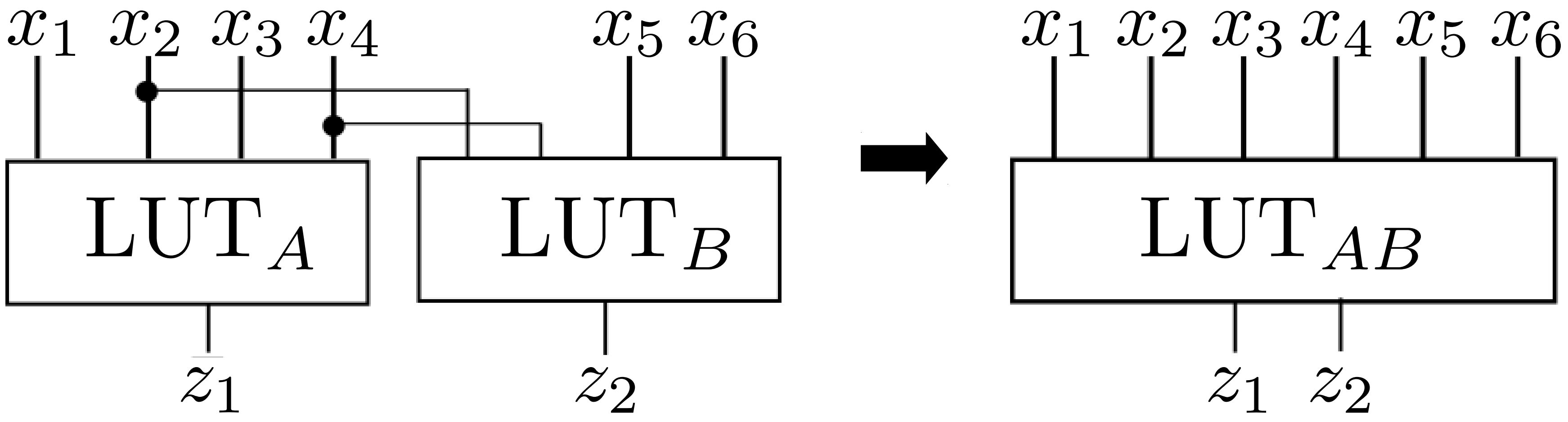
Add post-processing to improve efficiency



Combining LUTs

FPGAs only support single output LUTs

We combine LUTs with similar inputs to improve efficiency



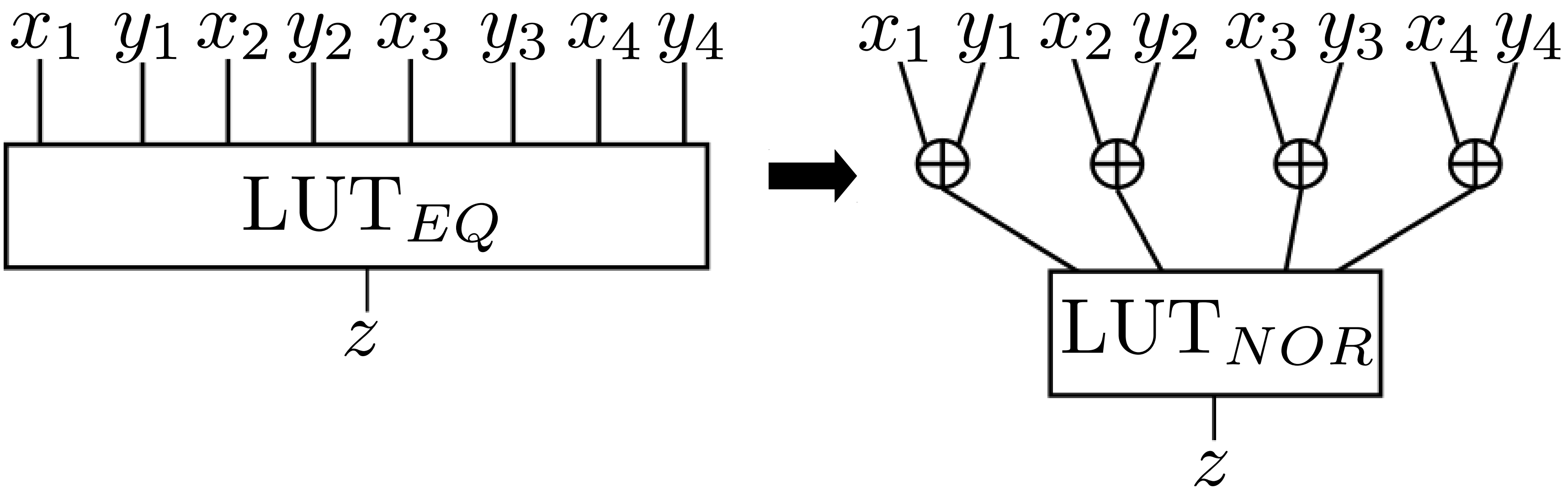
SP-LUT Communication: 512 bits

SP-LUT Communication: 380 bits

Extracting XORs

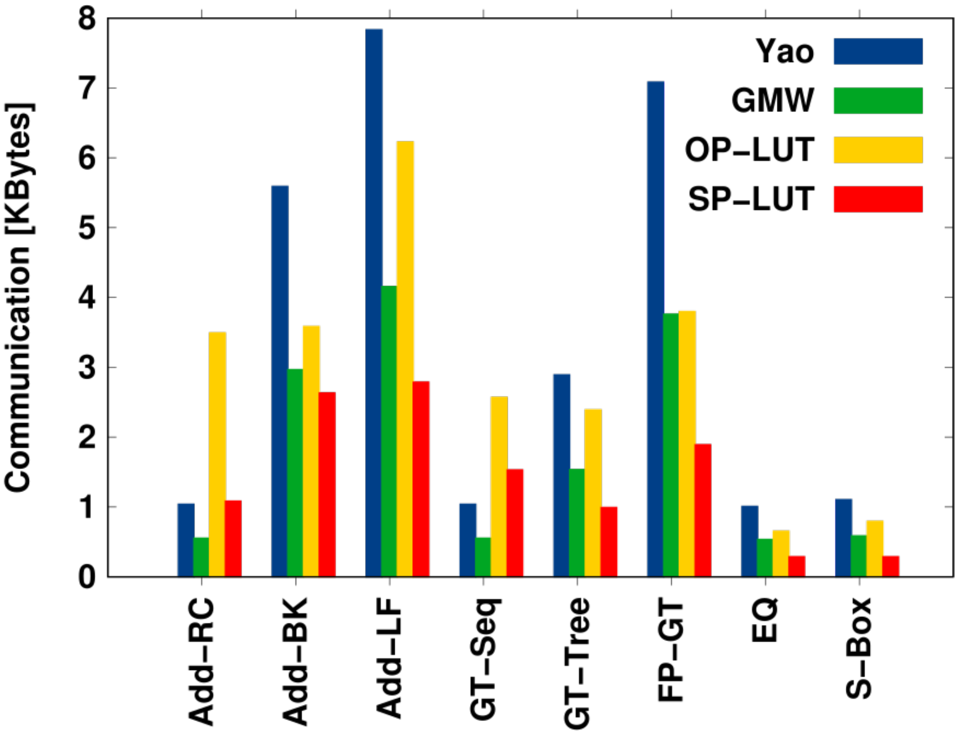
Since XORs are free, we can extract them

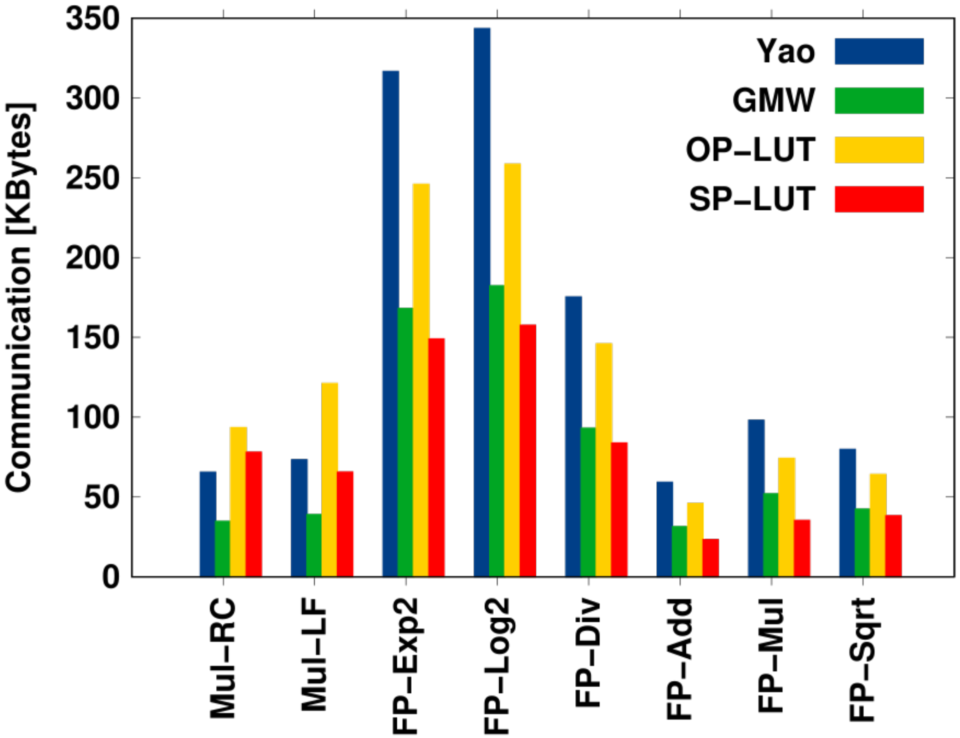
Example $z = (x \overset{?}{=} y)$


x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4
LUT_{EQ}
z
x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4
$\oplus$
LUT_{NOR}
z

Comparison

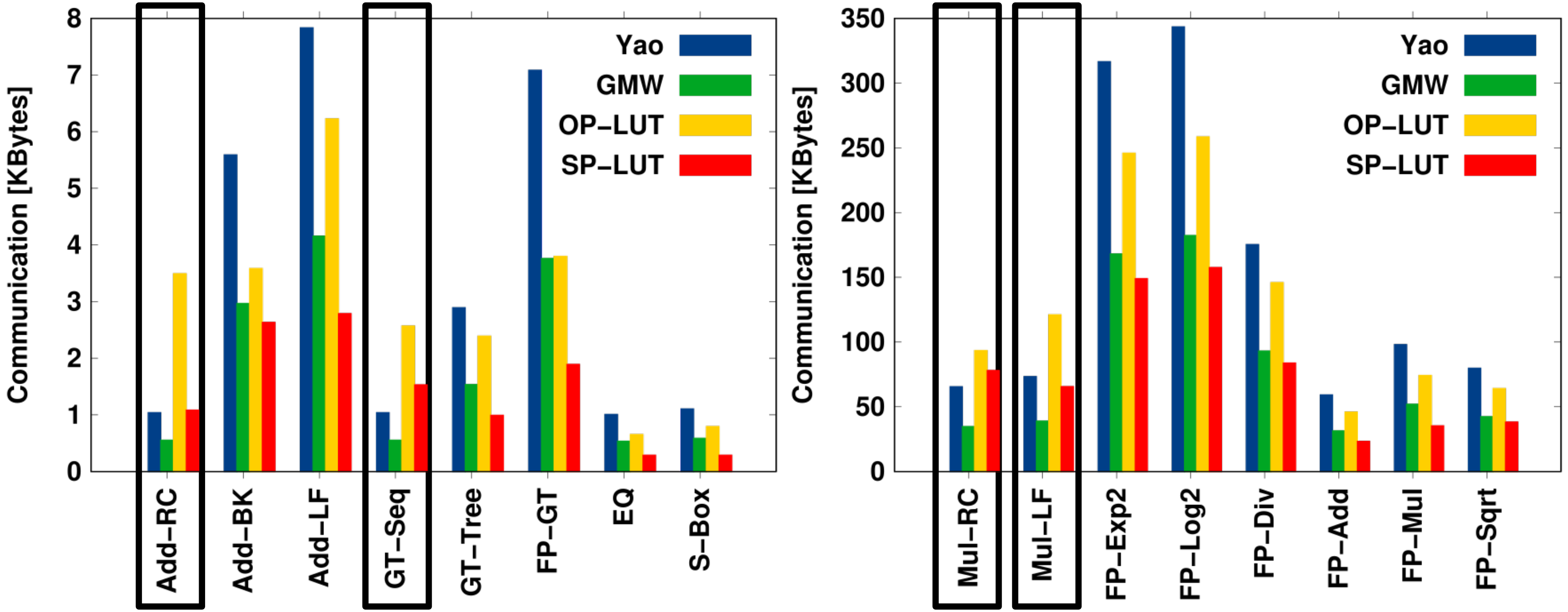
Communication





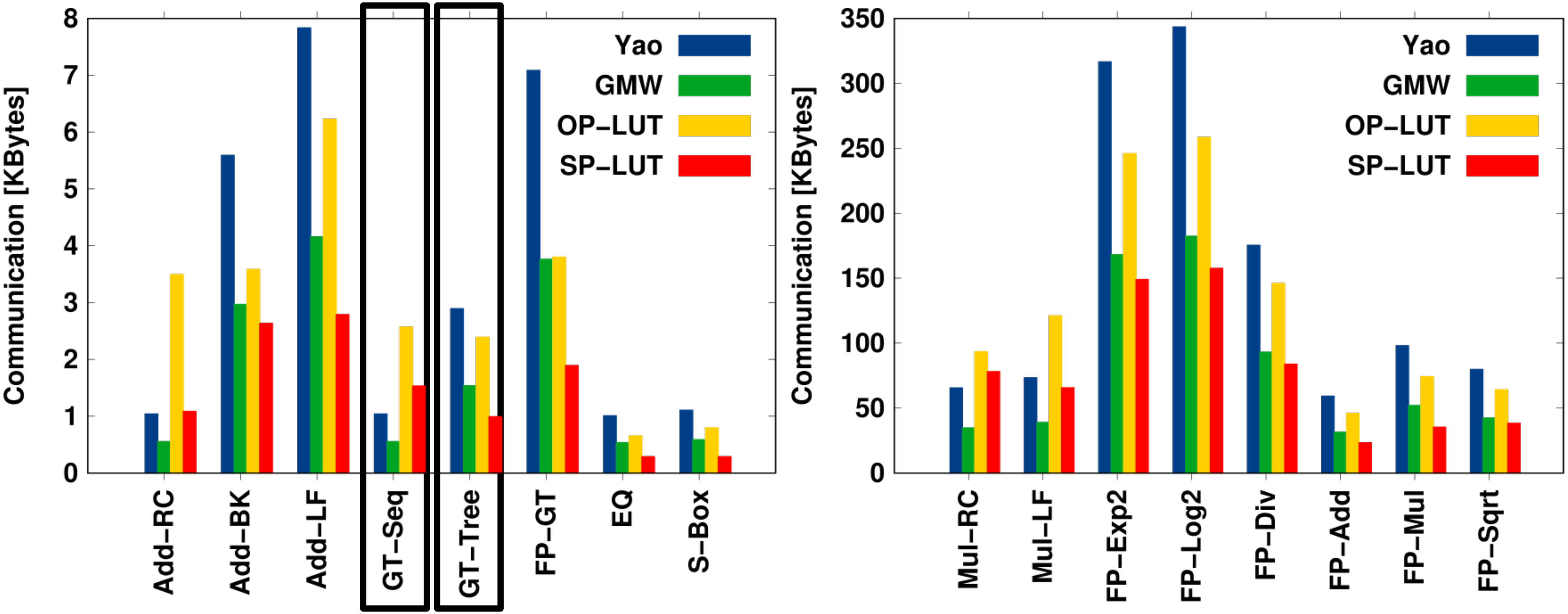
- Mostly: SP-LUT < GMW < OP-LUT < Yao
- Boolean circuits perform better for sequential structures
- LUT circuits perform best for tree based structures

Communication



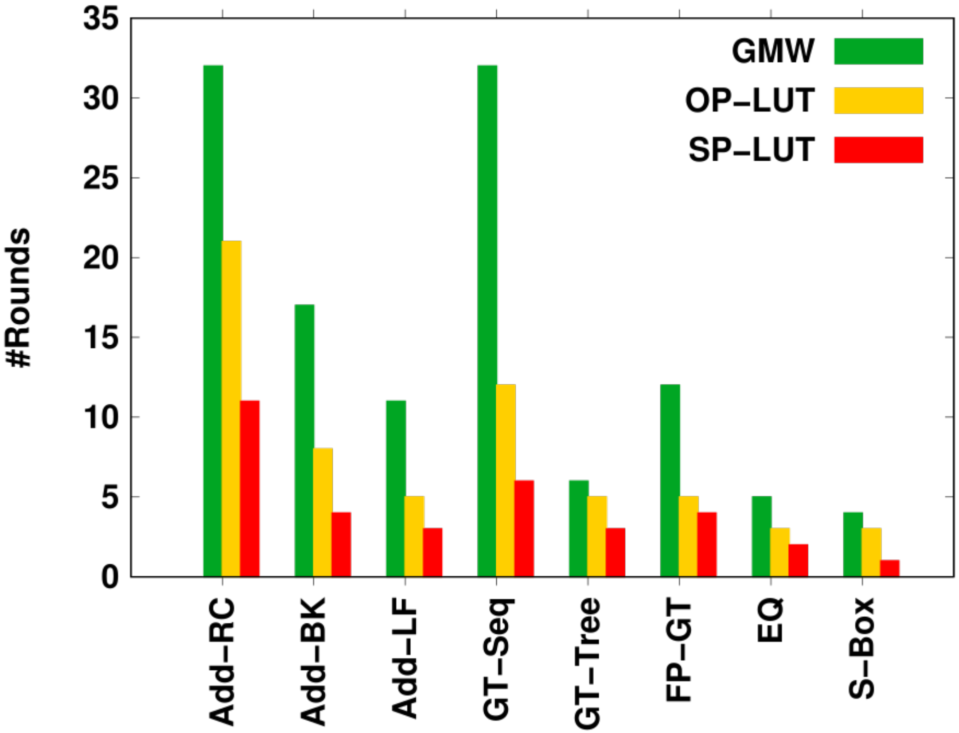
- Mostly: SP-LUT < GMW < OP-LUT < Yao
- Boolean circuits perform better for sequential structures
- LUT circuits perform best for tree based structures

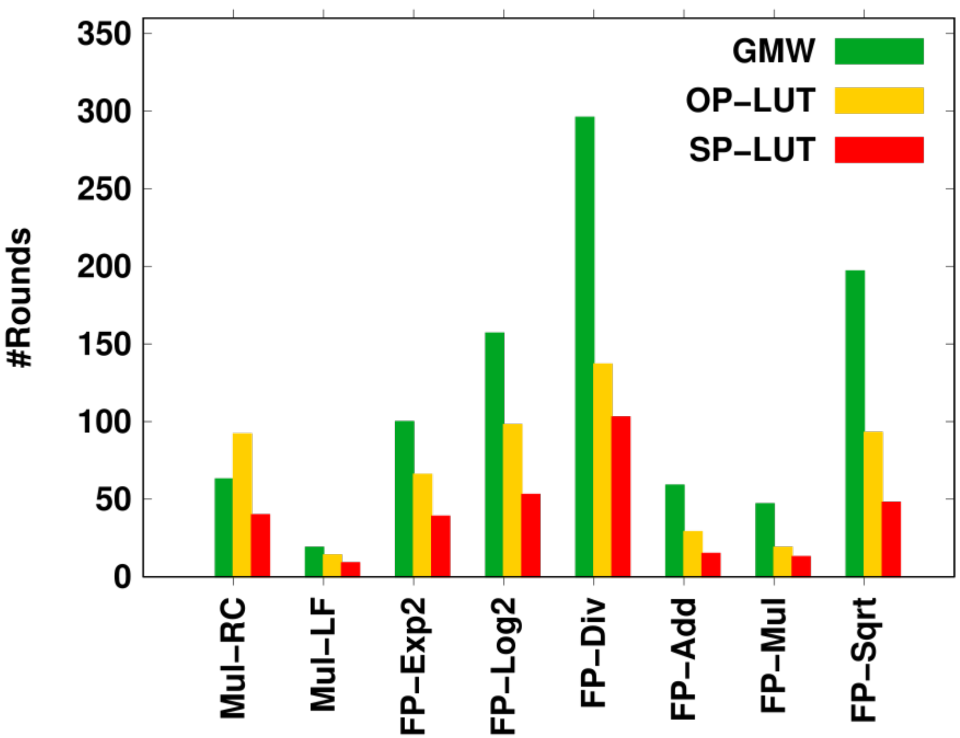
Communication



- Mostly: SP-LUT < GMW < OP-LUT < Yao
- Boolean circuits perform better for sequential structures
- LUT circuits perform best for tree based structures

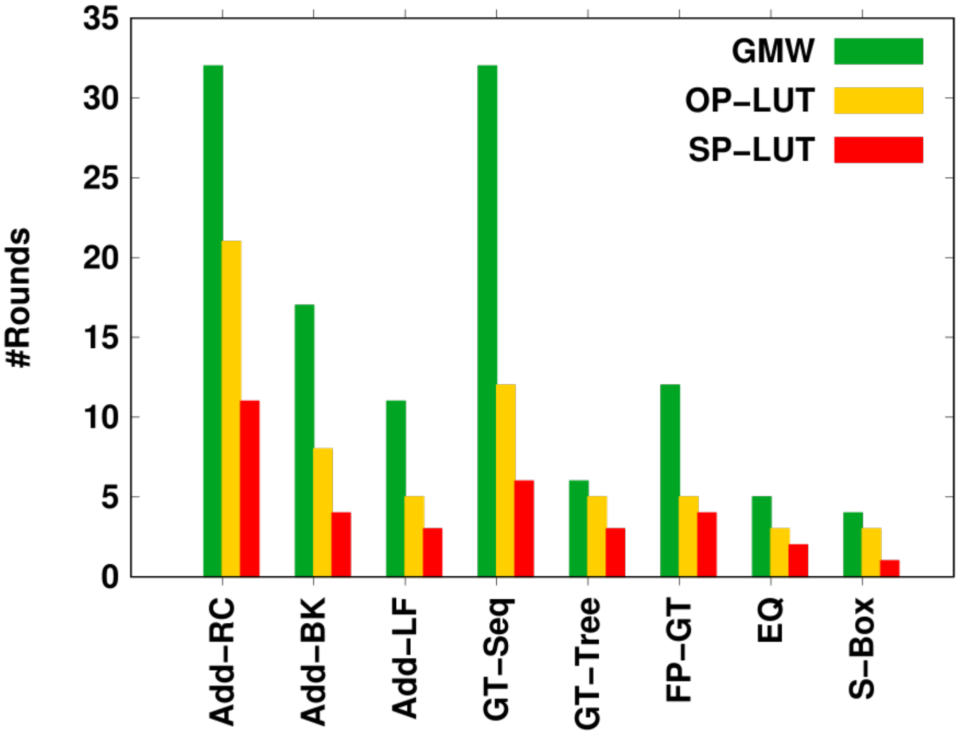
Interaction Rounds

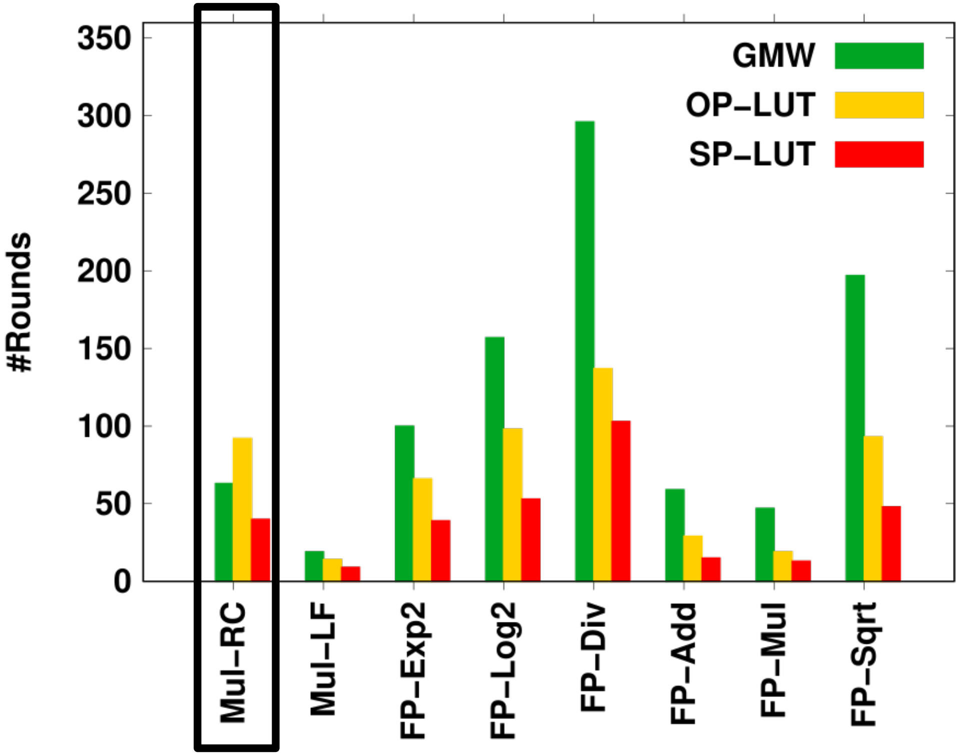




- Yao is constant round
- Mostly: SP-LUT < OP-LUT < GMW
- Exception: Multiplication with Ripple-carry addition

Interaction Rounds



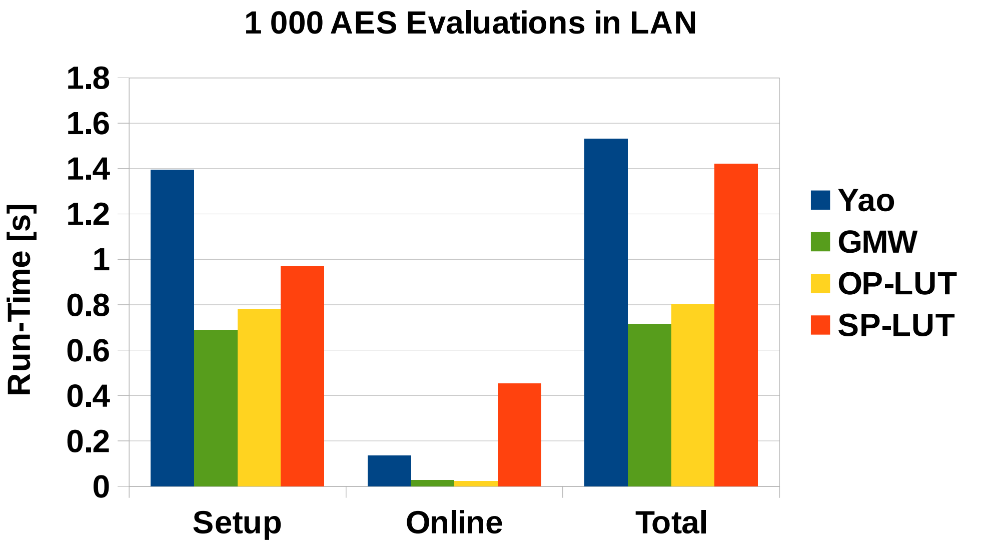


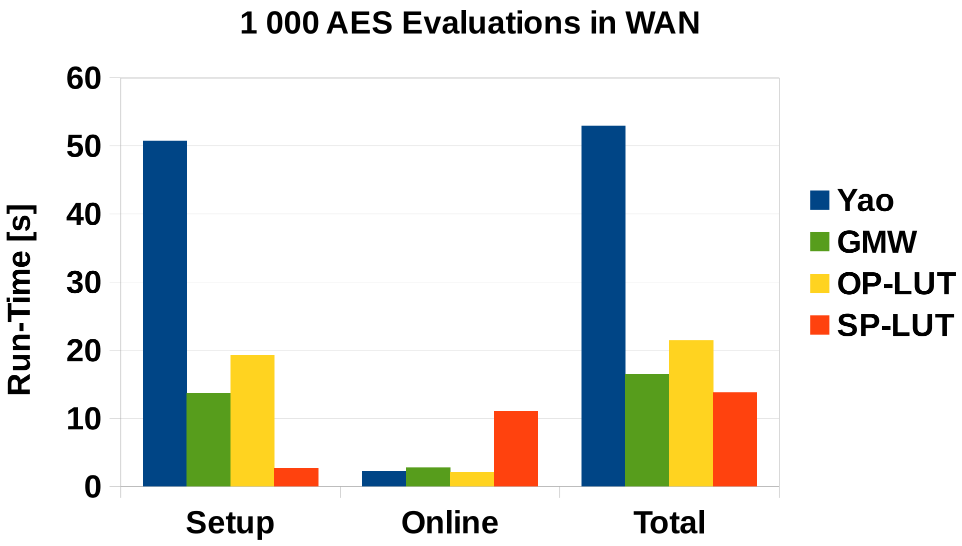
- Yao is constant round
- Mostly: SP-LUT < OP-LUT < GMW
- Exception: Multiplication with Ripple-carry addition

Empirical Evaluation

AES encryption of 1000 blocks using 4 threads

- LAN (1 GBit network, 0.2 ms latency)
- WAN (28 MBit network, 122ms latency)





Conclusion

Communication is bottleneck in 2PC

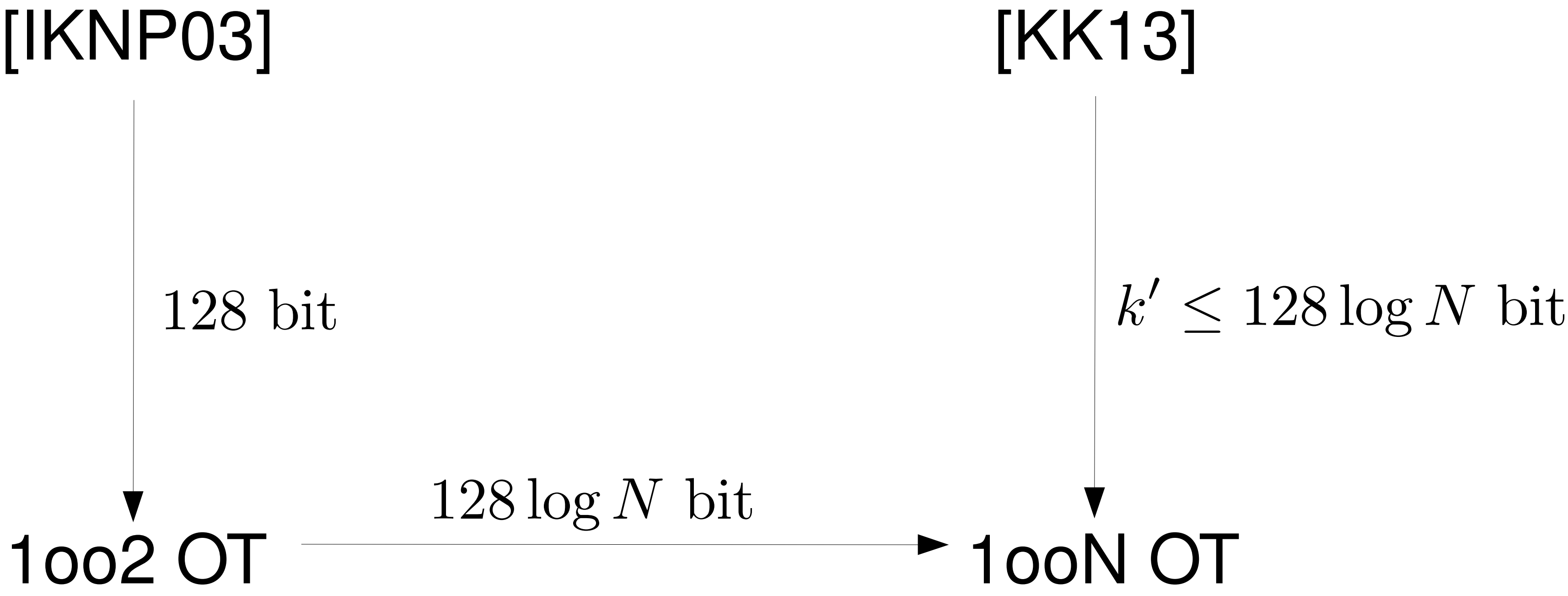
Developed LUT protocols based on 1ooN OT

Tool chain for compiling LUT circuits

Showed that LUT protocols can improve communication

Thank you for your attention

From 1oo2 OT to 1ooN OT



[IKNP03]

128 bit

1oo2 OT

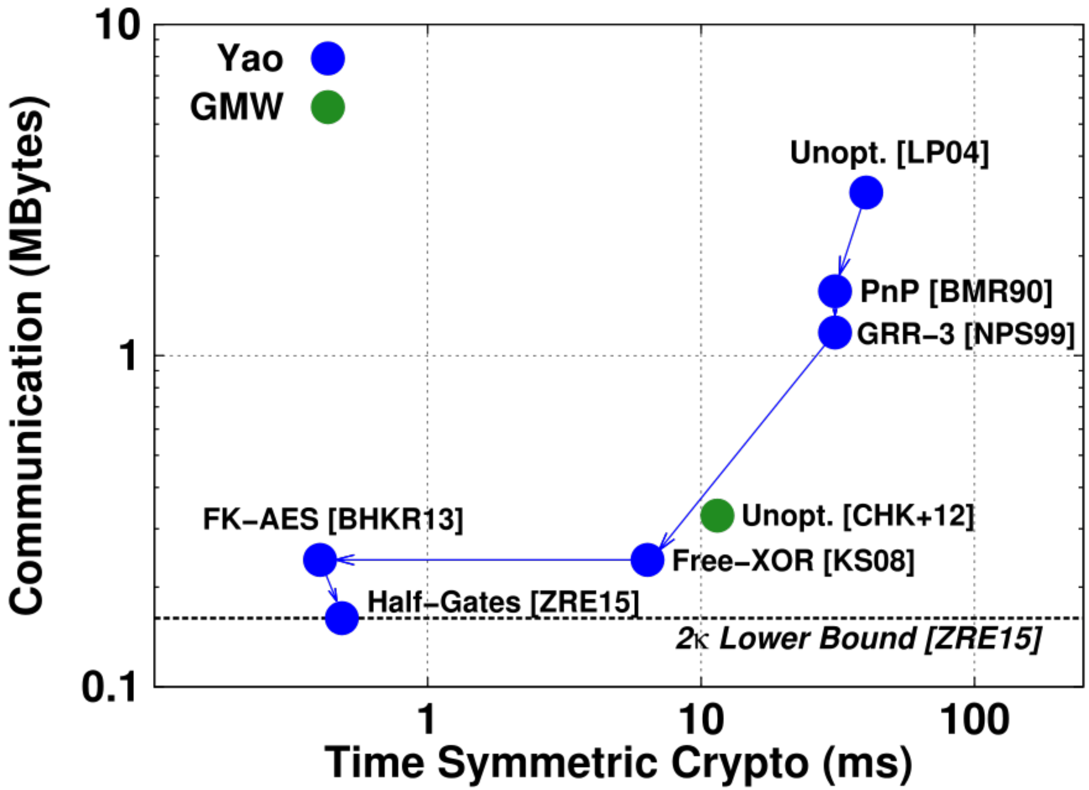
$128 \log N \text{ bit}$

[KK13]

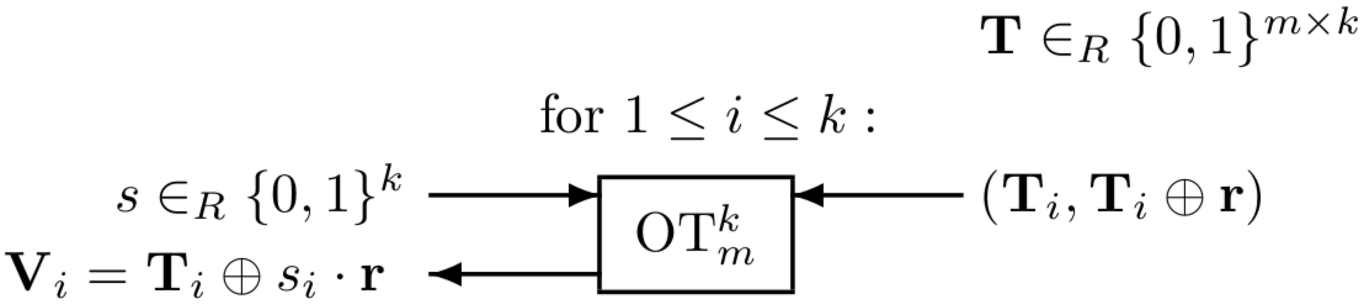
$k' \leq 128 \log N \text{ bit}$

1ooN OT

Our Results


Yao ●
GMW ●
Unopt. [LP04]
PnP [BMR90]
GRR–3 [NPS99]
Unopt. [CHK+12]
FK–AES [BHKR13]
Free–XOR [KS08]
Half–Gates [ZRE15]
2κ Lower Bound [ZRE15]
10
1
0.1
Communication (MBytes)
1
10
100
Time Symmetric Crypto (ms)

1ooN OT Extension [KK13]



$\mathbf{T} \in_R \{0,1\}^{m \times k}$

for $1 \leq i \leq k$:

$s \in_R \{0,1\}^k$

$\mathbf{V}_i = \mathbf{T}_i \oplus s_i \cdot \mathbf{r}$

OT_m^k

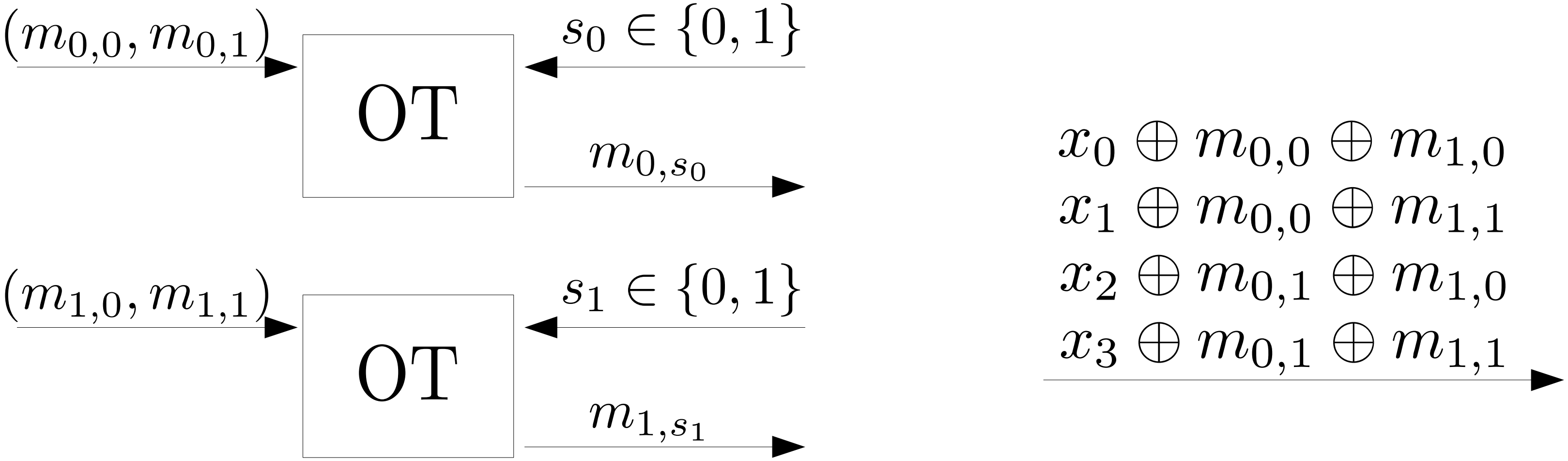
$(\mathbf{T}_i, \mathbf{T}_i \oplus \mathbf{r})$

$$k \updownarrow \underbrace{\begin{array}{cc} \text{if } r_j = 0 & \text{if } r_j = 1 \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ . \\ . \\ 0 \end{pmatrix} & \begin{pmatrix} 1 \\ 1 \\ 1 \\ . \\ . \\ 1 \end{pmatrix} \end{array}}_{\text{Hamming distance } k} \Rightarrow k' \updownarrow \underbrace{\begin{array}{ccccc} \text{if } r_j = 0 & \text{if } r_j = 1 & \text{if } r_j = 2 & \cdots & \text{if } r_j = N-1 \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ . \\ . \\ 1 \end{pmatrix} & \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ . \\ . \\ 1 \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ . \\ . \\ 0 \end{pmatrix} & \begin{matrix} \cdots \\ \\ \\ \\ \cdots \end{matrix} & \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ . \\ . \\ 0 \end{pmatrix} \end{array}}_{\text{Codewords with HD } k}$$

$$k' \leq 128 \log N$$

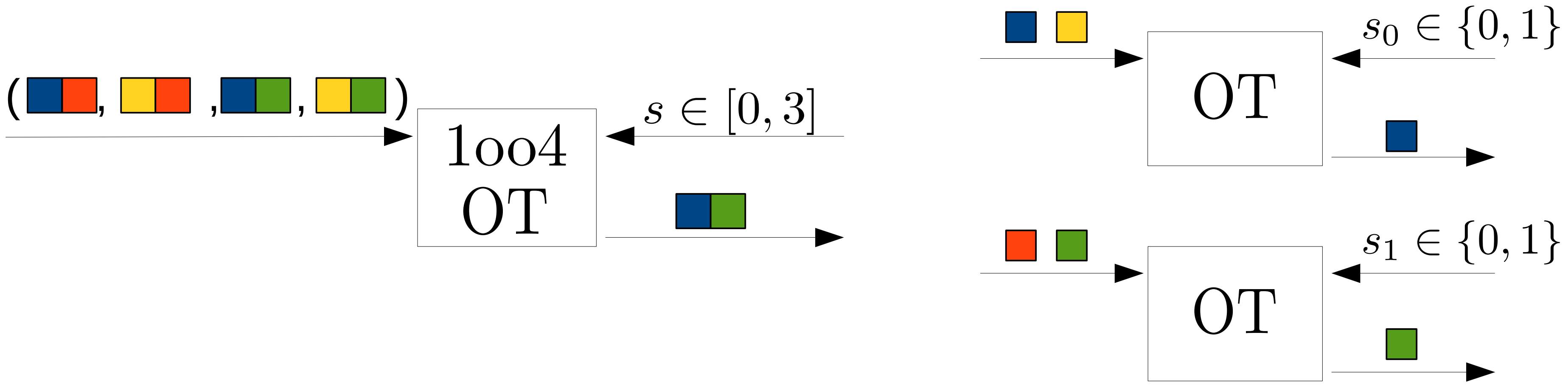
From 1oo2 OT to 1ooN OT

- 1ooN OT can be obtained from logN 1oo2 OTs
- Example 1oo4:


$(m_{0,0}, m_{0,1})$
OT
$s_0 \in \{0,1\}$
m_{0,s_0}
$(m_{1,0}, m_{1,1})$
OT
$s_1 \in \{0,1\}$
m_{1,s_1}
$x_0 \oplus m_{0,0} \oplus m_{1,0}$
$x_1 \oplus m_{0,0} \oplus m_{1,1}$
$x_2 \oplus m_{0,1} \oplus m_{1,0}$
$x_3 \oplus m_{0,1} \oplus m_{1,1}$

From 1ooN OT to 1oo2 OT

- Surprising insight: reducing 1ooN OT to single bit 1oo2 OT saves communication



- Best for N=16: Requires 320 bits instead of 512 bits