Efficient OT Extension and its Impact on Secure Computation

Pushing the Communication Barrier of Passive Secure Two-Party Computation

TECHNISCHE UNIVERSITÄT DARMSTADT

Michael Zohner (TU Darmstadt)

Joint work with
Ghada Dessouky, Ahmad-Reza Sadeghi, Thomas Schneider,
Shaza Zeitouni (all TU Darmstadt)
Farinaz Koushanfar (UC San Diego)

EC SPRIDE

Applications

Auctions, ...

Private Set Intersection, ...

Machine Learning, ...

Biometric Identification, ...

This work: passive security and security parameter k = 128 bit

Secure Two-Party Computation Protocols

Yao's garbled circuits protocol

- Function-dependent setup phase
- Constant round
- ≥ 256 bit communication per AND (simplex)

Very fast implementations

- Fairplay ~1 000 Gates/s
- FastGC ~100 000 AND/s
- ObliVM ~3 million AND/s — *Passive*

- Billion Gate ~100 000 AND/s — *Active*
- Blazing Fast ~500 000 AND/s
- More blazing fast

Secure Two-Party Computation Protocols

GMW
- Function-independent setup phase
- ≥ 256 bit communication per AND (duplex)

Setup Phase: pre-compute multiplication triples (MTs) using OT
- ApricOT: passive=active ~7 million OTs/s
- ABY: ~3 million AND/s *Passive*
- TinyOT: ~400 000 AND/s *Active*
- SPDZ: 5 000 Mult/s of 128 bit values

Online Phase: simple computation and small messages but multi-round

Status Quo

Good news: extremely fast computation
- JustGarble generates ~2GBit/s traffic per thread
- Passive OT extension generates ~1 Gbit/s traffic per thread

Bad news: communication boundary
- LAN connection provides 1Gbit/s
- Lowerbound on linear garbling schemes [ZRE15]
- Online time of GMW very latency dependent

Computation resources scale better than communication

Bottleneck: Communication and round complexity

Related Work

[KK13] outlines efficient 1ooN OT extension variant

+ Reduces communication per AND in GMW from 256 to 160 bit
- High computation overhead

[IKMOP13,DZ16,TinyTable] uses multi-input tables for secure computation

+ Reduces communication and rounds in the online phase
- High setup costs

GESS [KK12] multi-round information-theoretic variant of garbled circuits

+ Reduces communication
- Unsure how to extend to arbitrary functionalities

What Did We Do?

TECHNISCHE UNIVERSITÄT DARMSTADT

1) Less communication for GMW
- Further optimize 1ooN OT extension of [KK13] to compute MTs
- Reduces communication from 256 to 134 bit per AND

2) Less communication and rounds Lookup Table (LUT) representation
- Online LUT (O-LUT): efficient online phase
- Setup LUT (S-LUT): efficient overall evaluation

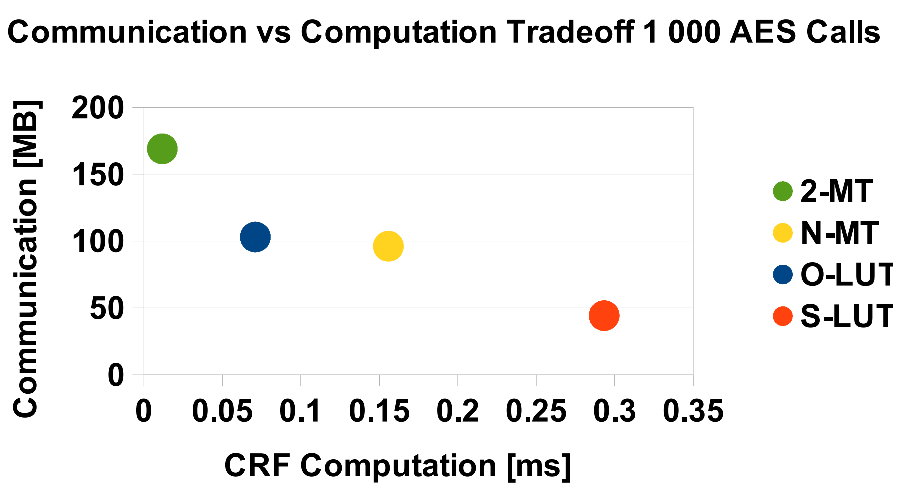
3) Tool support for generating LUT representations

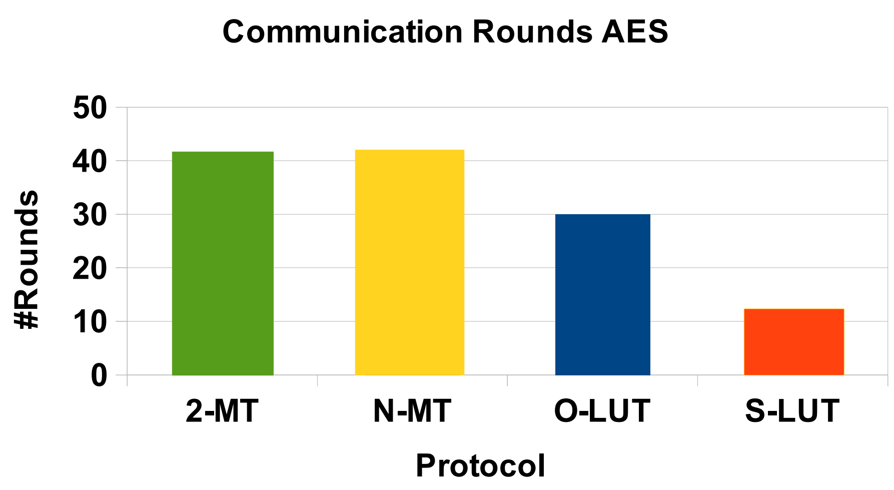
4) Evaluation on various basic operations

Our Results

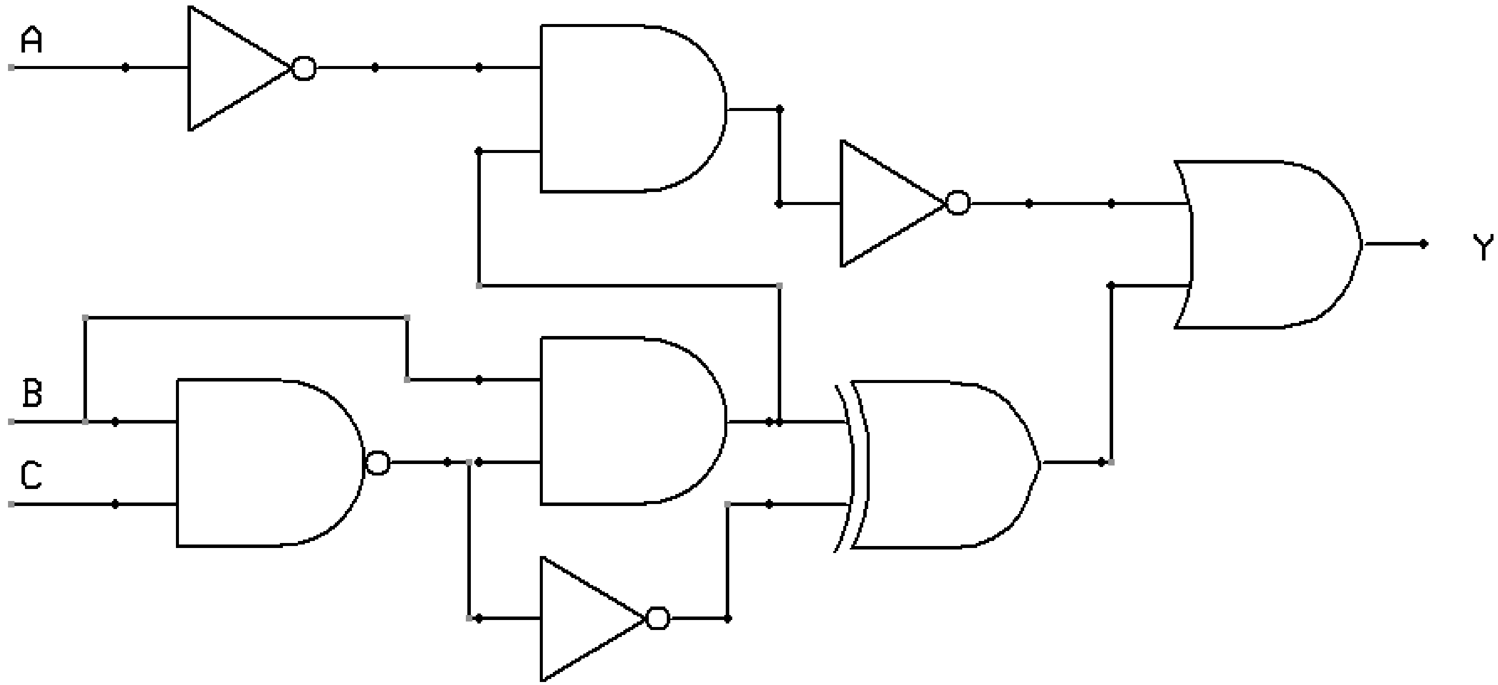
Trade more computation for 1) less communication and 2) less rounds

- 2-MT: GMW from 1-out-of-2 OT extension
- N-MT: GMW from 1-out-of-N OT extension [KK13]
- O-LUT: LUT protocol with efficient online phase
- S-LUT: LUT protocol with better overall communication





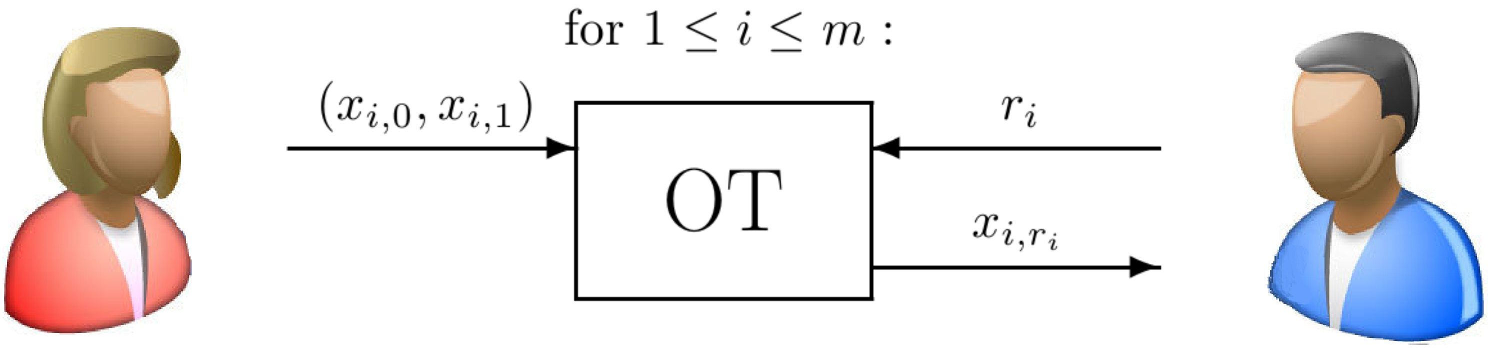
Part 1) Less Communication for GMW



1oo2 OT Extension [IKNP03]

Alice holds m pairs of messages $(x_{i,0}, x_{i,1})$

Bob holds m-bit string r and wants to obtain x_{i,r_i} in i-th OT



1oo2 OT Extension [IKNP03] (Base-OT Step)

TECHNISCHE UNIVERSITÄT DARMSTADT

Alice and Bob switch roles and perform *k* base OTs

$$(x_{j,0}, x_{j,1}) \in \{0,1\}^{2\ell}$$

$$\mathbf{r} = (r_1, ..., r_m) \in \{0,1\}^m$$

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

$$\text{for } 1 \leq i \leq k:$$

$$s \in_R \{0,1\}^k \longrightarrow \boxed{\mathrm{OT}_m^k} \longleftarrow (\mathbf{T}_i, \mathbf{T}_i \oplus \mathbf{r})$$

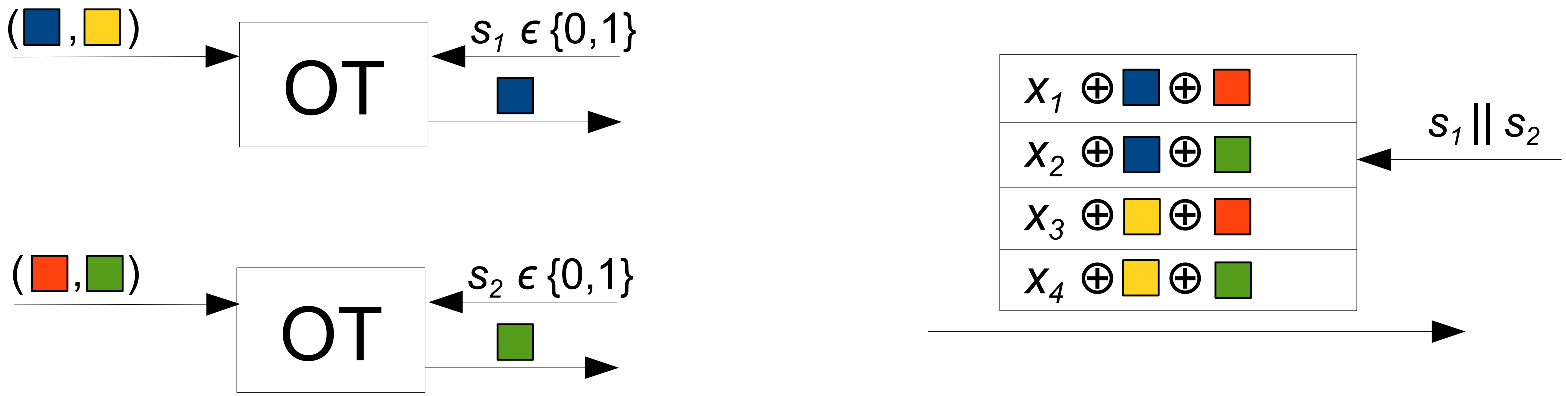
$$\mathbf{V}_i = \mathbf{T}_i \oplus s_i \cdot \mathbf{r} \longleftarrow \boxed{\mathrm{OT}_m^k}$$

From 1oo2 OT to 1ooN OT

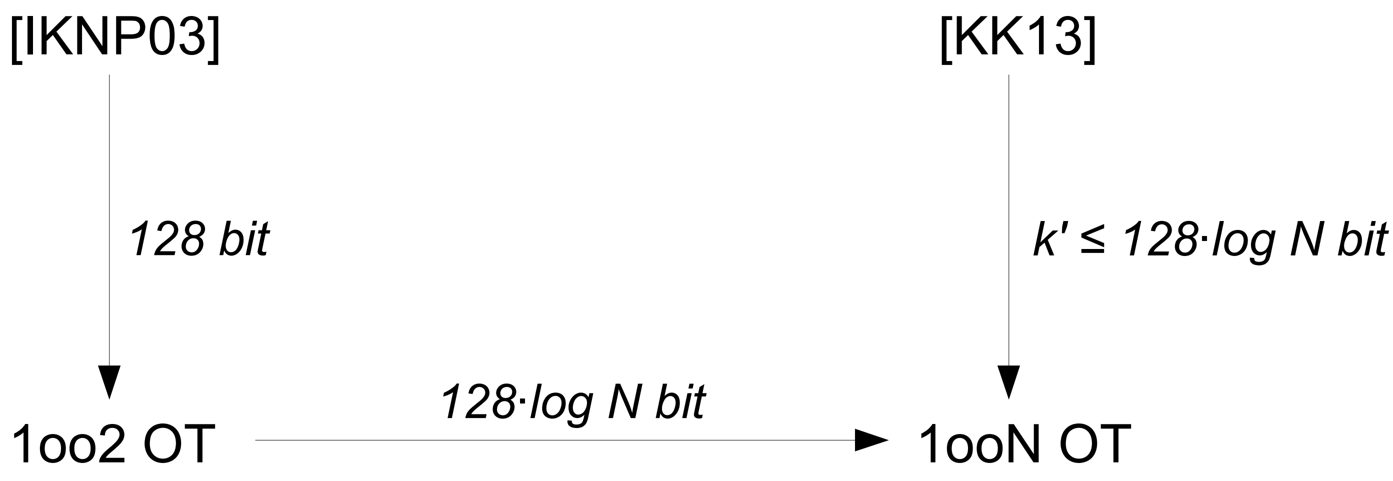
From 1oo2 OT to 1ooN OT

1ooN OT can be obtained from *log N* invocations of 1oo2 OT extension

Example: 1oo4 OT for $(x_1,\ldots,x_4)$

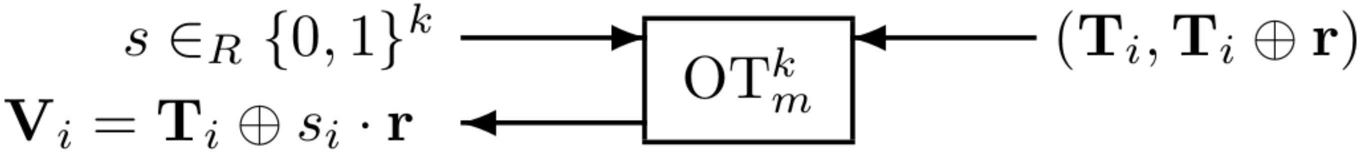


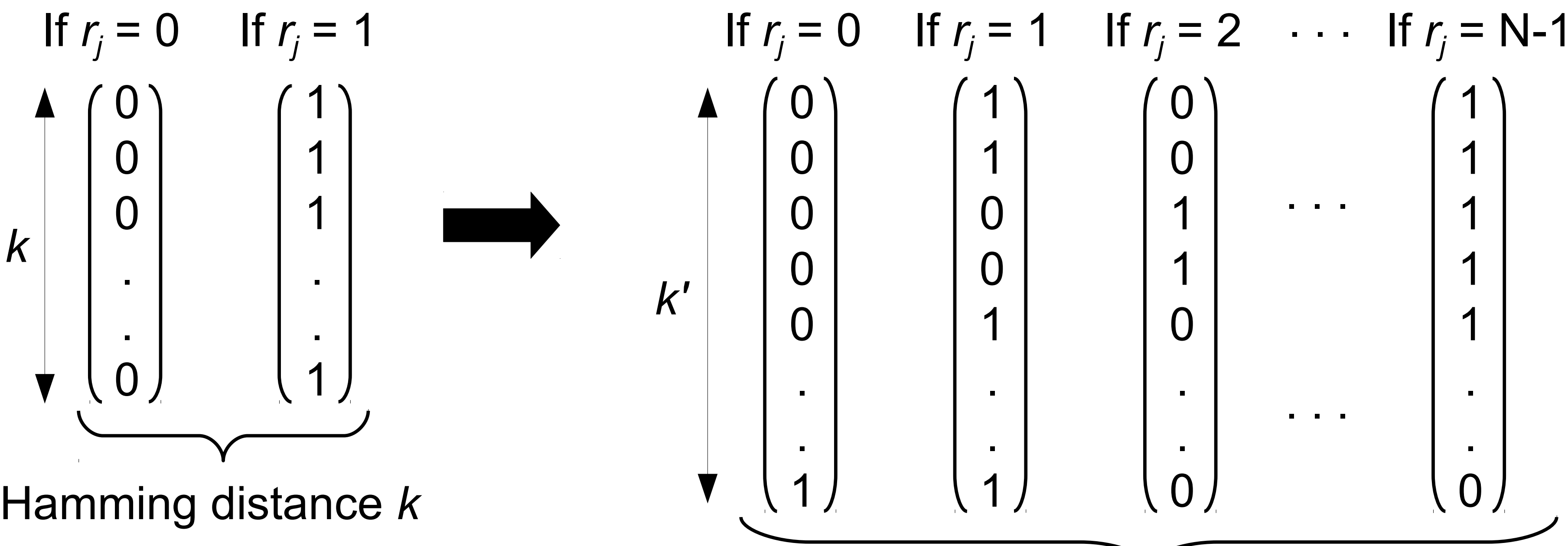
1ooN OT Extension [KK13]



Generalization to 1ooN OT Extension [KK13]

for $1 \le i \le k$:





$k' \le 128 \log N$

Codewords with HD k

1ooN OT Extension [KK13] (Efficiency)

TECHNISCHE UNIVERSITÄT DARMSTADT

The codewords need *k* bit Hamming distance (HD)

Efficiency of the [KK13] 1ooN OT depends on the underlying code

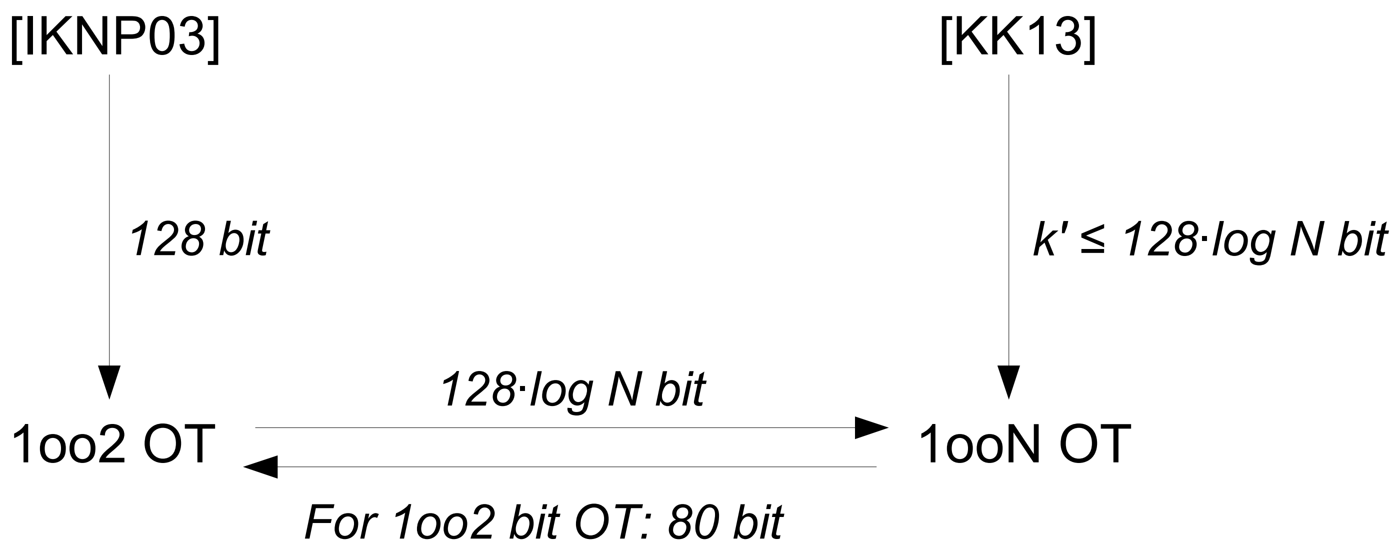
For $N = 2$: use repetition code
- Same as the [IKNP03] protocol

For $2 < N \leq 2k$: use a Walsh Hadamard code
- *h* codewords with *h* bit length and HD $h/2$
- Since we require HD=k we have $2k$=256 bit codewords

For $N > 2k$: use linear codes
- Achieves $O(k)$ communication instead of $O(k \log N)$
- Concrete improvements for PSI on 128-bit elements

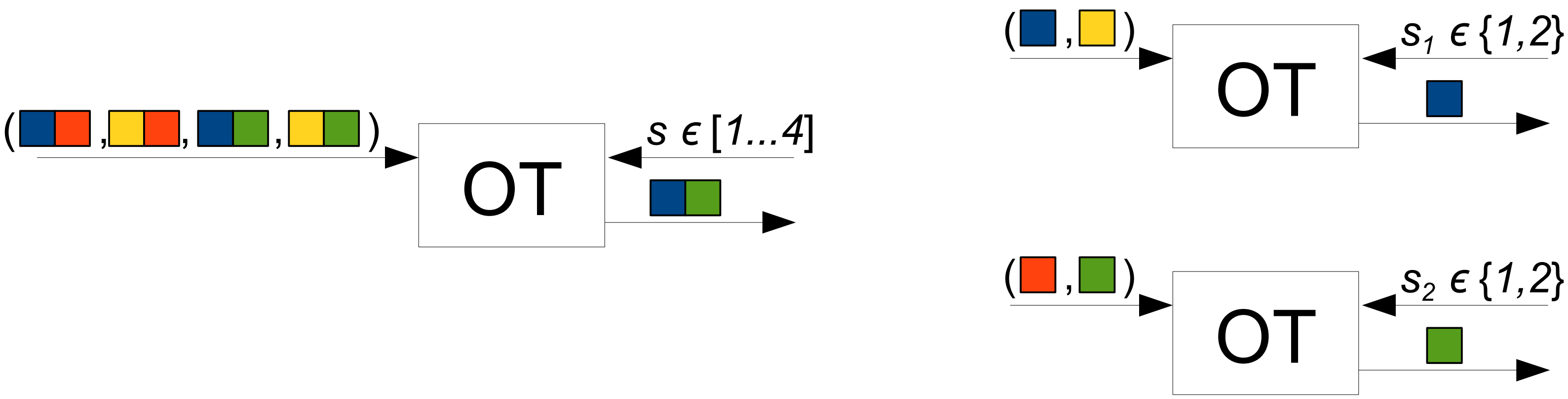
EC SPRIDE

1ooN OT Extension [KK13]



1ooN OT Extension for Short Strings [KK13]

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



Best for *N*=16: requires only 320 bits instead of 512 bits

[KK13] Downside: Increased Computation

1oo2 OT extension uses efficient fixed-key AES-128 [BHKR13]

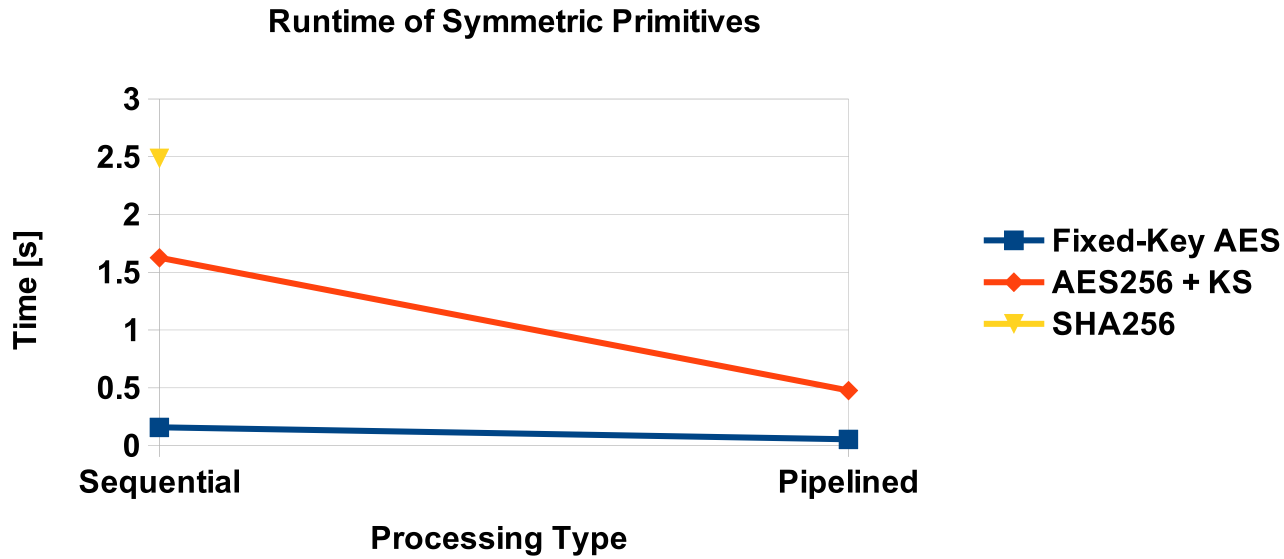
1ooN OT processes values with >128-bit length
- Too large for AES encryption
- Replace by AES-256 with key schedule [KSS12] → 30 times slower

Even worse: 1ooN OT needs N evaluations while 1oo2 OT needs 2*log N*
- For N=16: 60x more computation
- For N=256: 480x more computation

Optimization 1) Improve Computation

TECHNISCHE UNIVERSITÄT DARMSTADT

Idea: Use pipelining of [GNLP15] for AES-256+KS



AES-256+KS only 9 instead of 30 times slower than fixed-key AES-128

EC SPRIDE

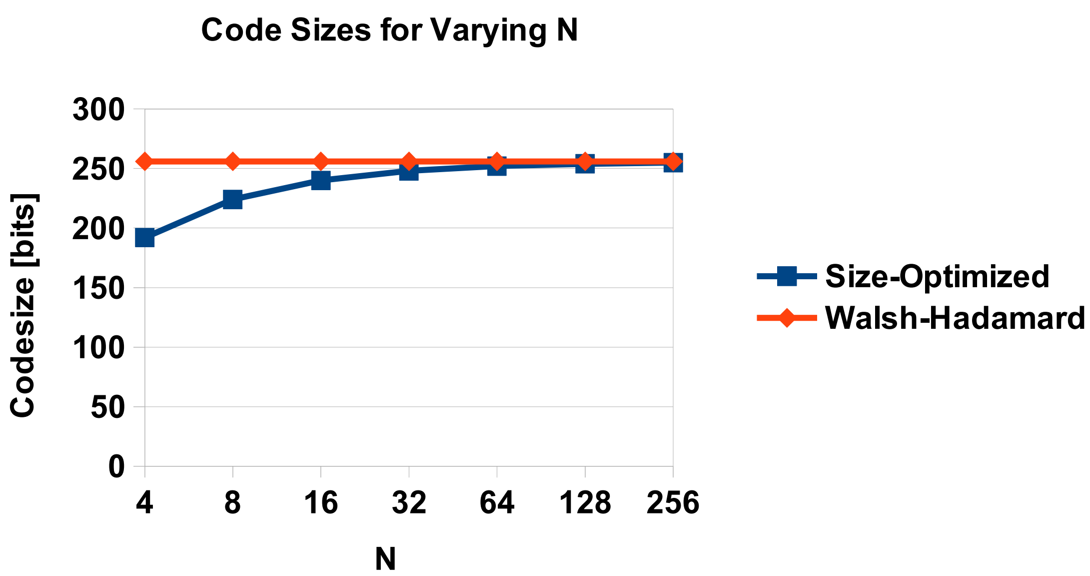
Optimization 2) Short Codes

For 2 < *N* < 2*k*, [KK13] uses Walsh-Hadamard code, which is not size-optimal

Improve communication using specific codes for specific *N*

- http://mint.sbg.ac.at/ gives short codes for different parameters

Saves between 25% and 1% communication

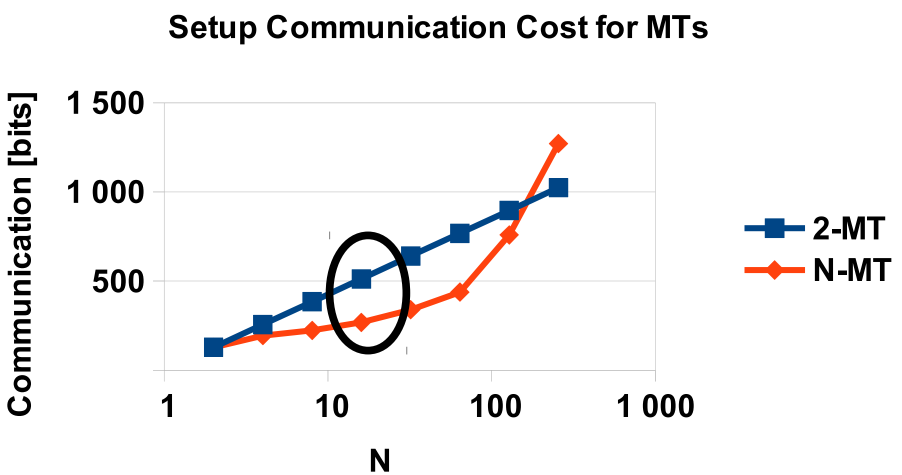


Optimization 3) MTs from 1ooN OT (N-MT)

TECHNISCHE UNIVERSITÄT DARMSTADT

[KK13] reduces 1ooN OT to 1oo2 OT for computing AND gates

Instead: reduce 1ooN OT to MTs (1oo4 OT)



Best for *N*=16: reducing communication from 256 to 134 bits per AND

Part 2) LUT-based Secure Computation

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

LUT Representation

Boolean circuits require one round per layer of AND gates

Process multi-input gates to decrease rounds [IKMOP13,DZ16]

- [IKMOP13] introduced one-time truth table (OTTT) protocol
- [DZ16] showed how to pre-compute OTTTs

OTTT [IKMOP13] Setup by Trusted Third Party

1) Represent as table

$f(x,y) = x+y$

y\x	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

2) Rotate table

r = 2

s = 1

y\x	0	1	2	3
0	5	6	3	4
1	2	3	0	1
2	3	4	1	2
3	4	5	2	3

=

3) Secret-Share

y\x	0	1	2	3
0	4	4	6	2
1	5	2	4	5
2	4	3	5	2
3	0	3	6	5

+

y\x	0	1	2	3
0	1	2	5	2
1	4	1	3	3
2	6	1	3	0
3	4	2	3	5

4) Distribute

M_0, r → P_0

M_1, s → P_1

OTTT (Online Phase)

Compute f(x,y)

P_0 (x=3, r=2, M_0)

u=(x+r)

y\x	0	1	2	3
0	4	4	6	2
1	5	2	4	5
2	4	3	5	2
3	0	3	6	5

P_1 (y=2, s=1, M_1)

v=(y+s)

y\x	0	1	2	3
0	1	2	5	2
1	4	1	3	3
2	6	1	3	0
3	4	2	3	5

u

v

$M_0[u,v]=3$

$M_1[u,v]=2$

Output f(x,y)=5

Output f(x,y)=5

Pre-Computing OTTTs via Circuits [DZ16]

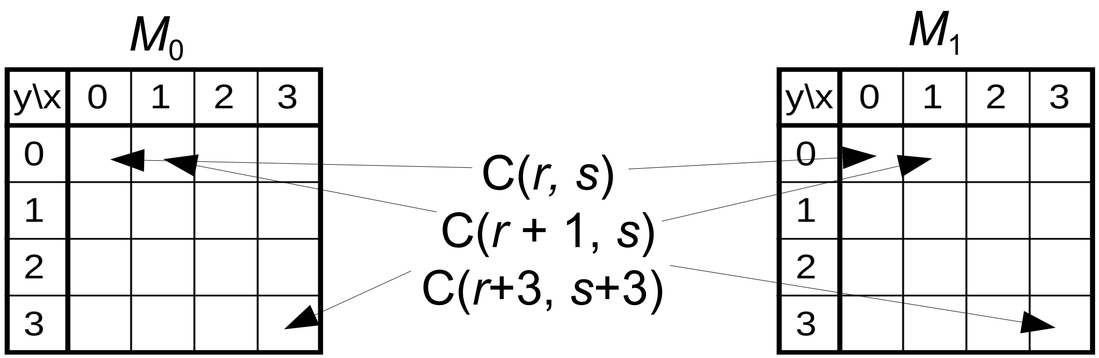
TECHNISCHE UNIVERSITÄT DARMSTADT

Use circuit-based protocols to pre-compute all table entries

1) Represent the function as circuit C

2) P_0 chooses random r, P_1 chooses random s

3) For all $0 \leq i, j \leq 3$, P_0 and P_1 evaluate $C(r+i, s+j) = (M_0[i,j], M_1[i,j])$



For circuits with δ input bits requires 2^δ evaluations with $\leq \delta$-1 ANDs

EC SPRIDE

LUT Protocols based on 1ooN OT

Circuit-based pre-computation of [DZ16] adds great overhead

- For δ-input LUTs 2^{δ} overhead compared to Boolean circuit

Idea: Use 1ooN OT to obliviously transfer OTTTs

- LUT communication becomes independent of the circuit cost

We outline two LUT protocols:

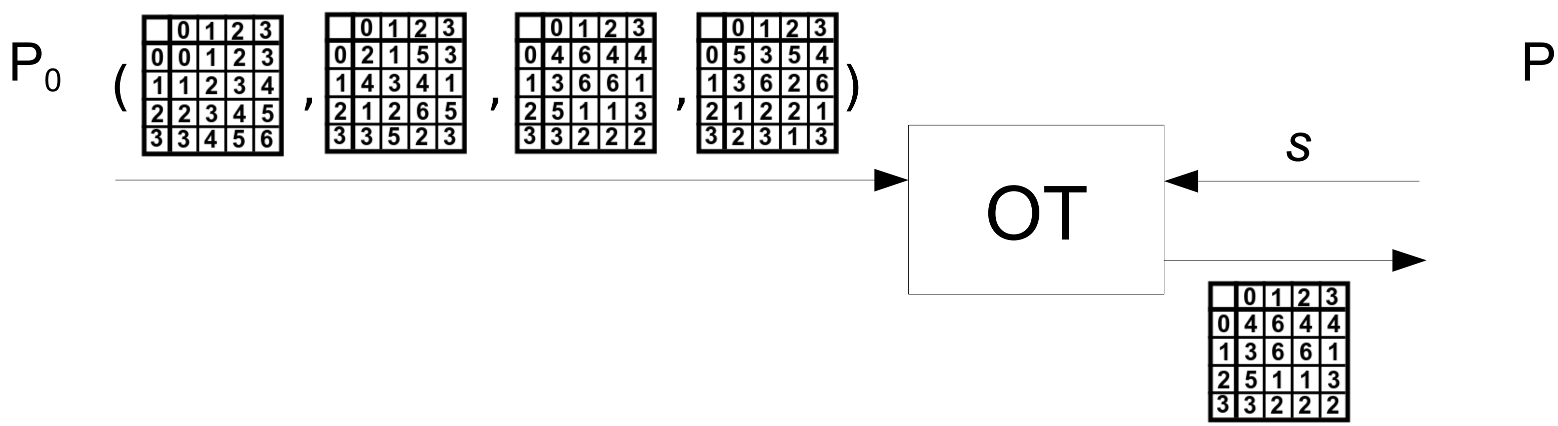
- Online LUT with low online communication
- Setup LUT with low overall but higher online communication

Online LUT (O-LUT)

Use 1ooN OT to transfer OTTTs for all possible choices of *s*

1) P_0 chooses random *r* and M_0 and prepares $M_{1,s'} = f(i+r, j+s') \oplus M_0$ for all $0 \leq i,j,s' \leq 3$

2) P_0 and P_1 perform a 1oo4 OT, where P_1 chooses a random table s



	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

	0	1	2	3
0	2	1	5	3
1	4	3	4	1
2	1	2	6	5
3	3	5	2	3

	0	1	2	3
0	4	6	4	4
1	3	6	6	1
2	5	1	1	3
3	3	2	2	2

	0	1	2	3
0	5	3	5	4
1	3	6	2	6
2	1	2	2	1
3	2	3	1	3

	0	1	2	3
0	4	6	4	4
1	3	6	6	1
2	5	1	1	3
3	3	2	2	2

For δ inputs the parties have to transfer 2^δ tables of 2^δ bits each

Setup LUT (S-LUT)

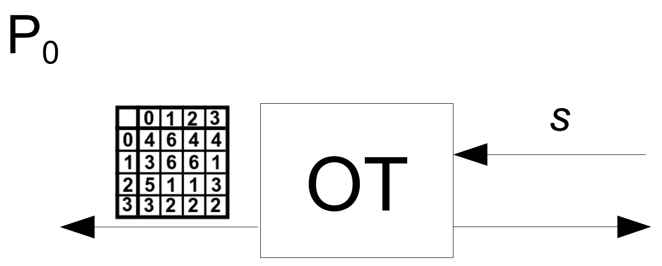
High setup communication for OTTTs with δ inputs

- Using circuits: between $138 \cdot 2^{\delta}$ and $(\delta\text{-}1) \cdot 138 \cdot 2^{\delta}$ bits
- Using 1ooN OT: $2^{2\delta}$ bits

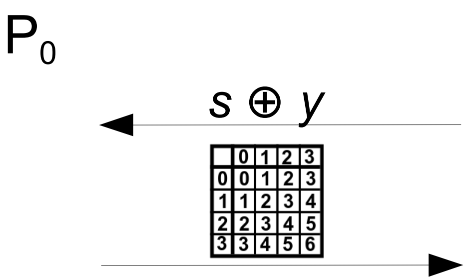
Problem: OTTTs are heavy since they require outputs for all possible inputs

Idea: pre-compute 1ooN OT in the setup phase and only update results in the online phase

Setup Phase

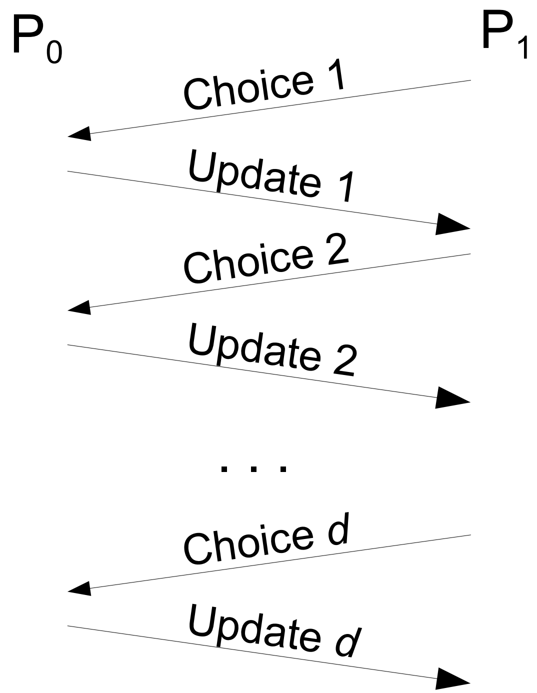


Online Phase



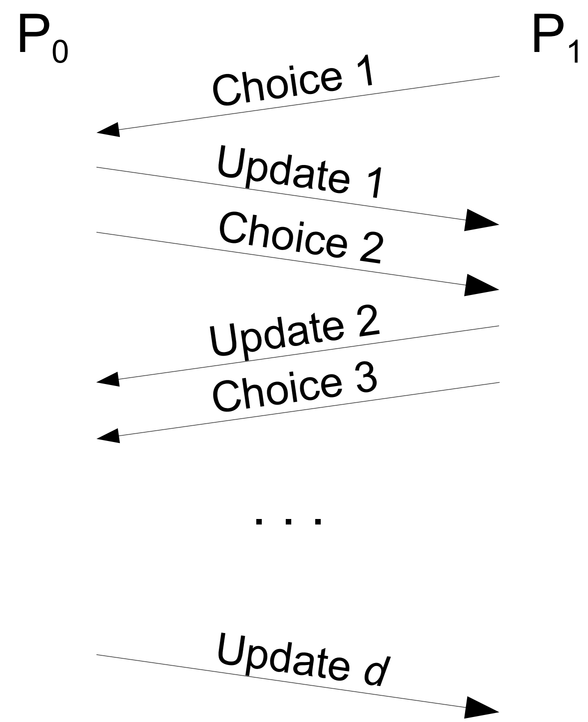
Improving S-LUT Round Complexity

OT Pre-computation



$2d$ rounds

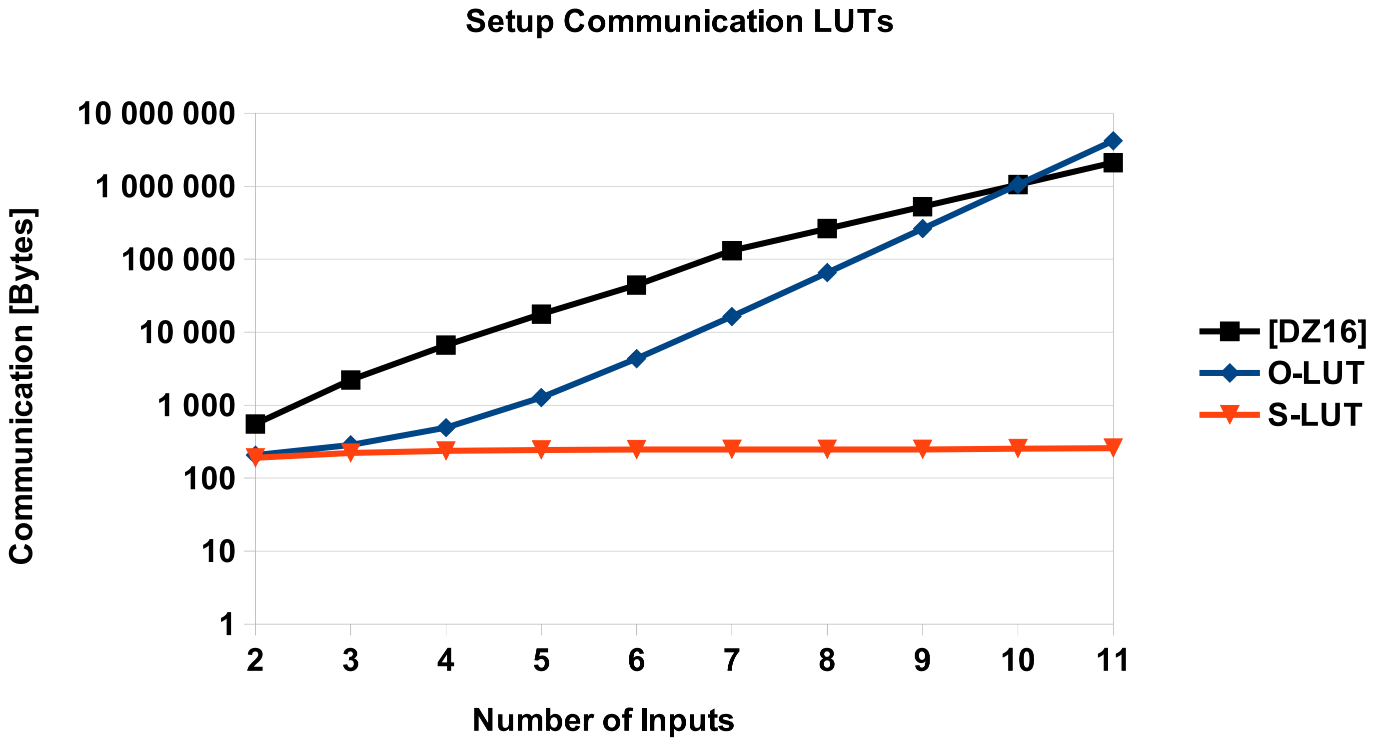
Role Switching [Huang12]



$d+1$ rounds

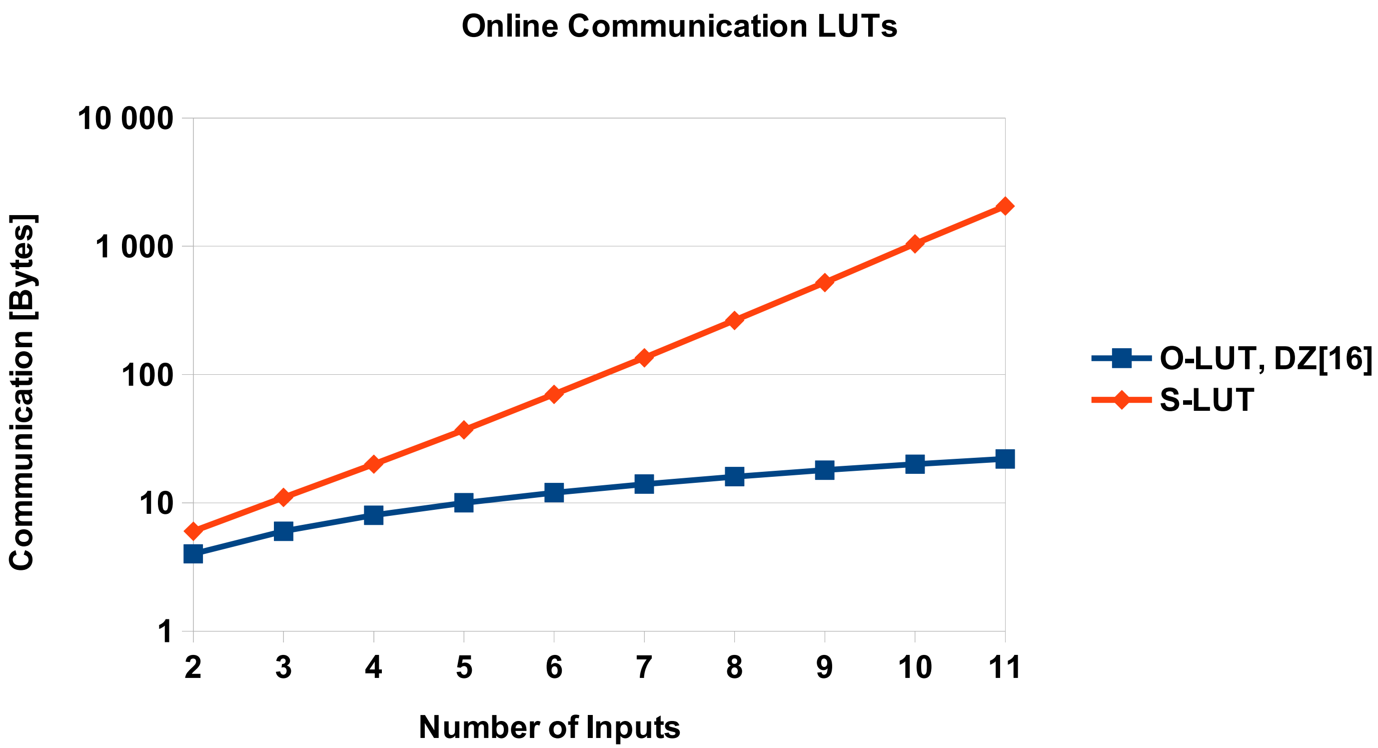
LUT Efficiency (Setup Phase)

TECHNISCHE UNIVERSITÄT DARMSTADT

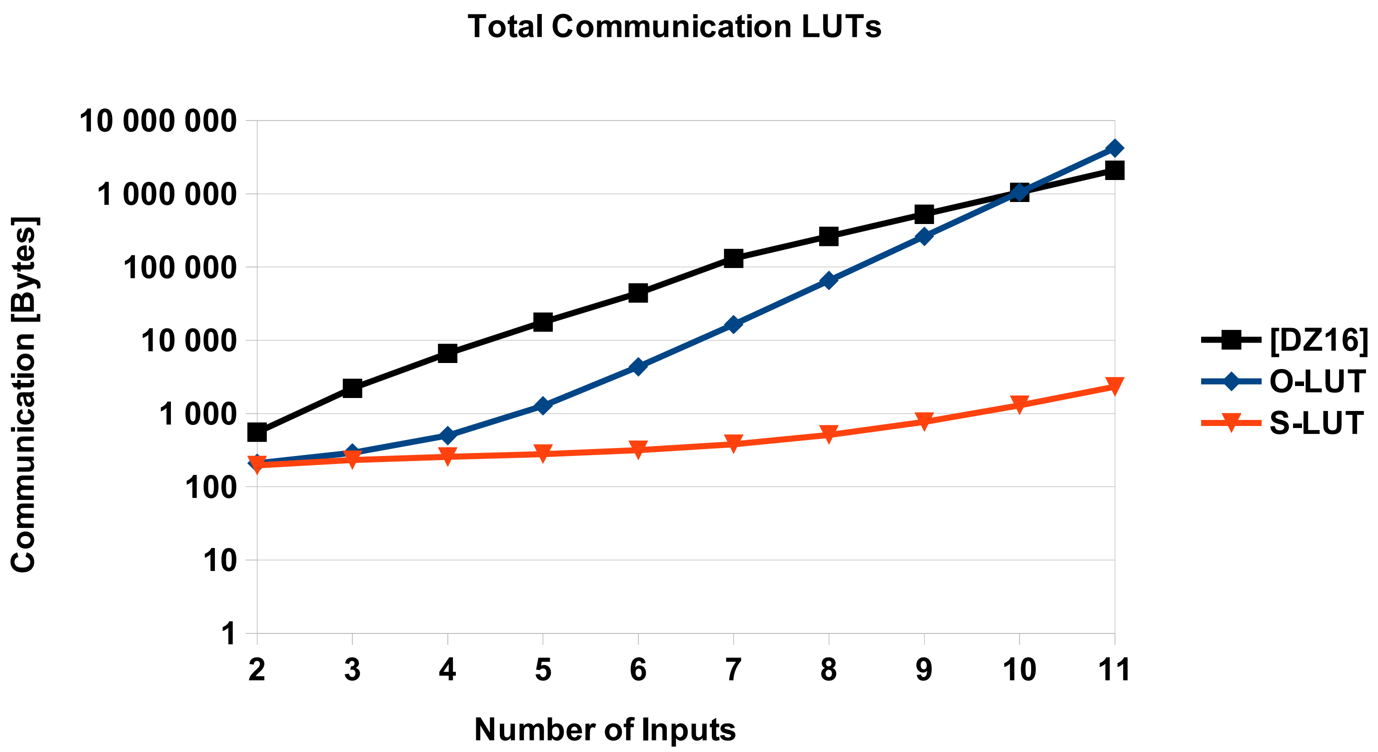


EC SPRIDE

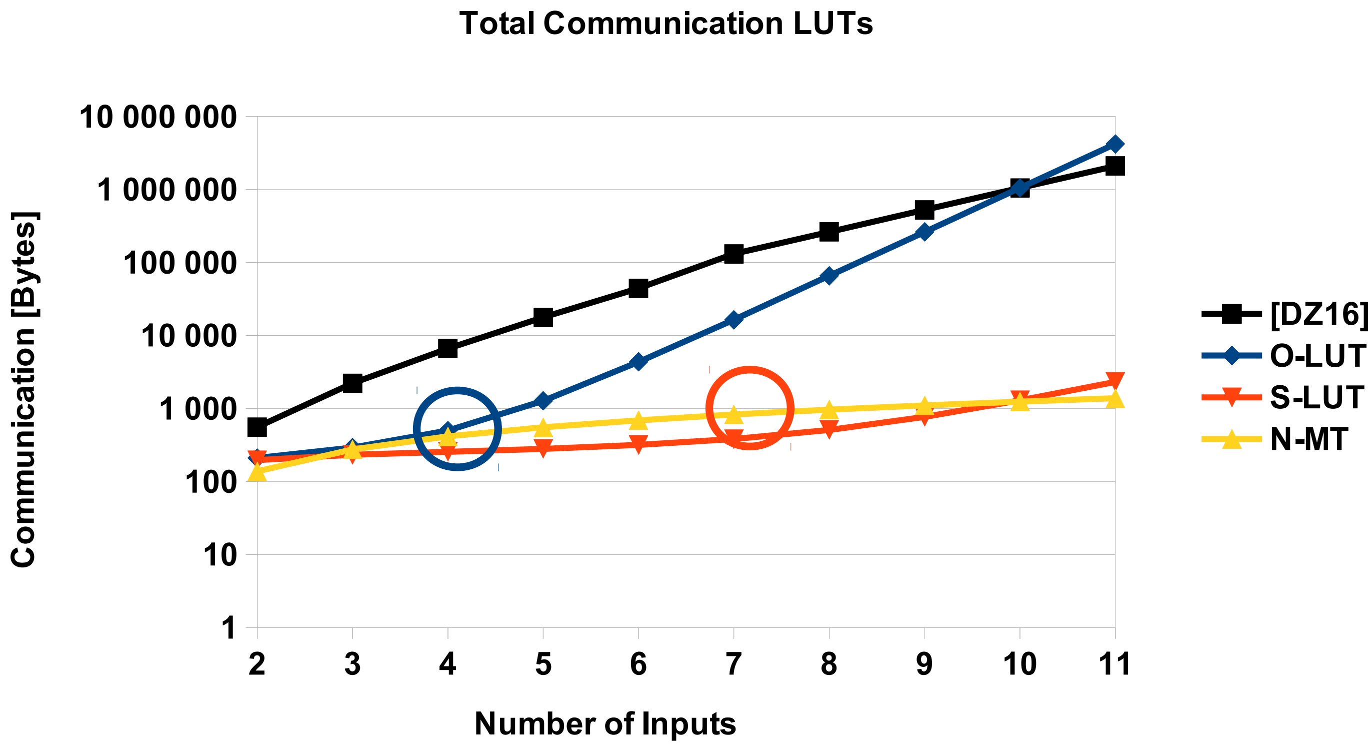
LUT Efficiency (Online Phase)



LUT Efficiency (Total)



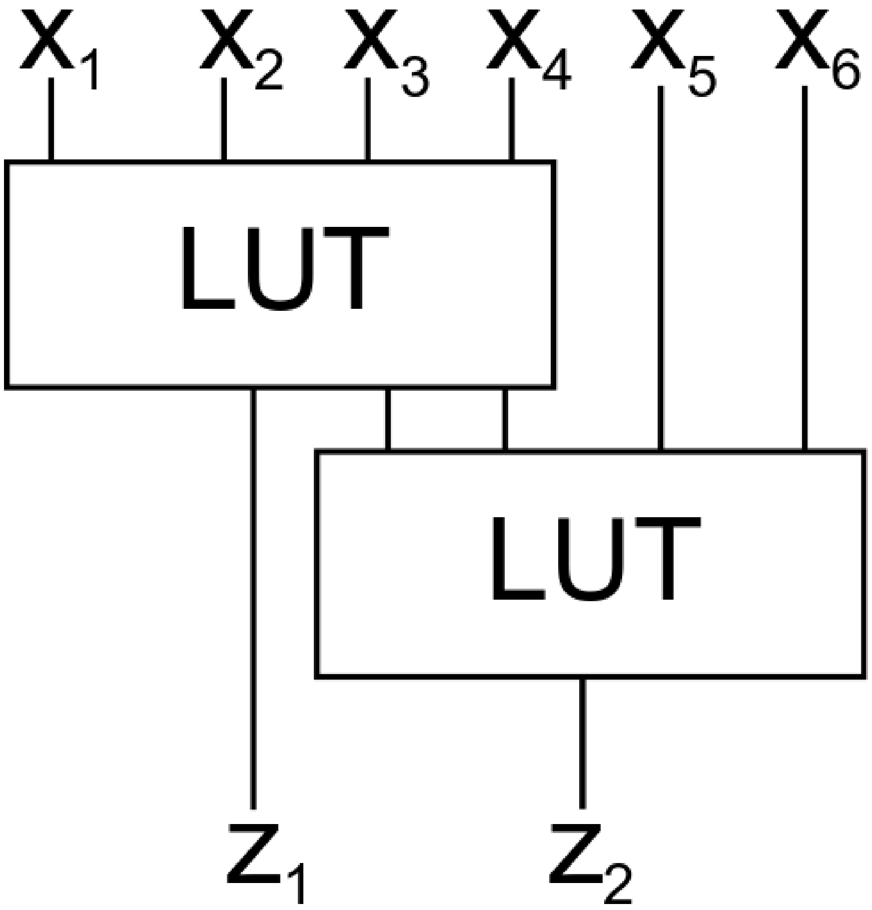
Communication vs Boolean Circuits



Optimum for O-LUT: 4 inputs (105% of N-MT)

Optimum for S-LUT: 7 inputs (45% of N-MT)

Part 3) Generating LUT Representations

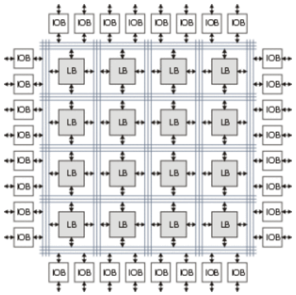


Yet Another Compiler?

Re-doing the work for LUTs is a time-consuming and error-prone task

=> Automate the generation of LUT representations

Idea: FPGAs internally operate on single output LUTs



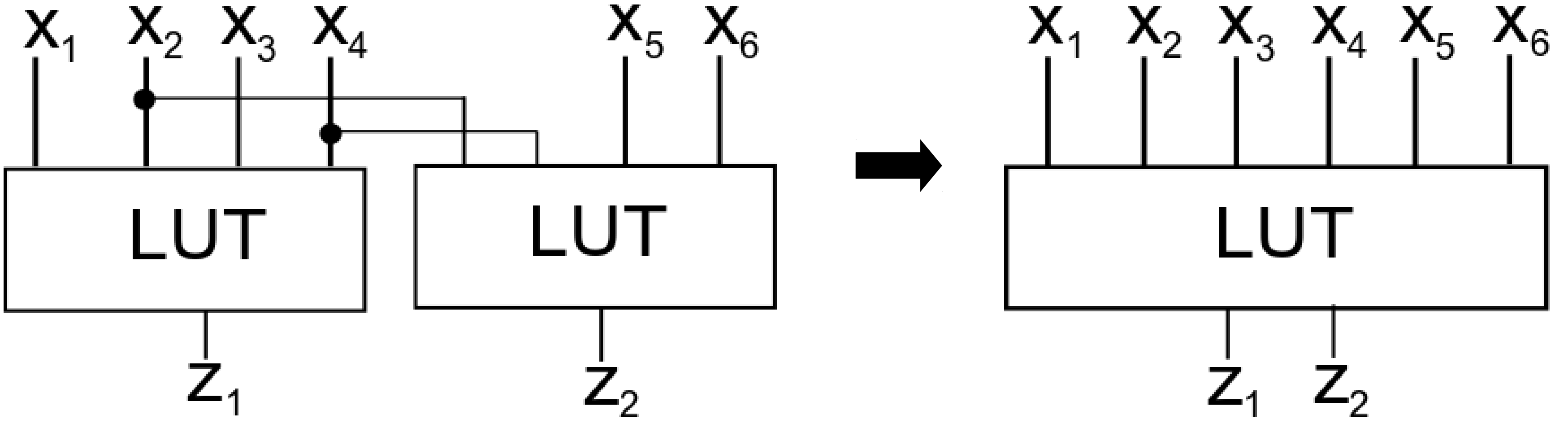
We use the ABC Logic synthesis tool to generate single output LUTs

Grouping Multi-Output LUTs

TECHNISCHE UNIVERSITÄT DARMSTADT

Problem: FPGAs only support LUTs with one output bit

We post-process and group LUTs with the same or similar inputs



S-LUT Communication: 512 bits

S-LUT Communication: 380 bits

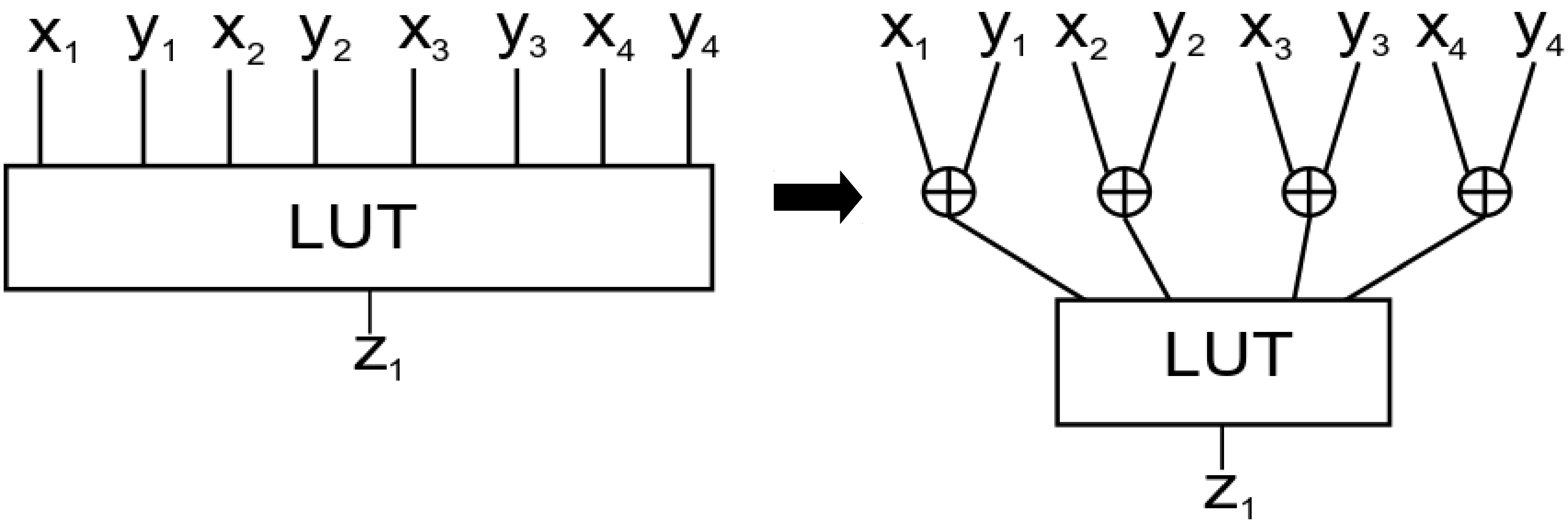
Extracting XORs

TECHNISCHE UNIVERSITÄT DARMSTADT

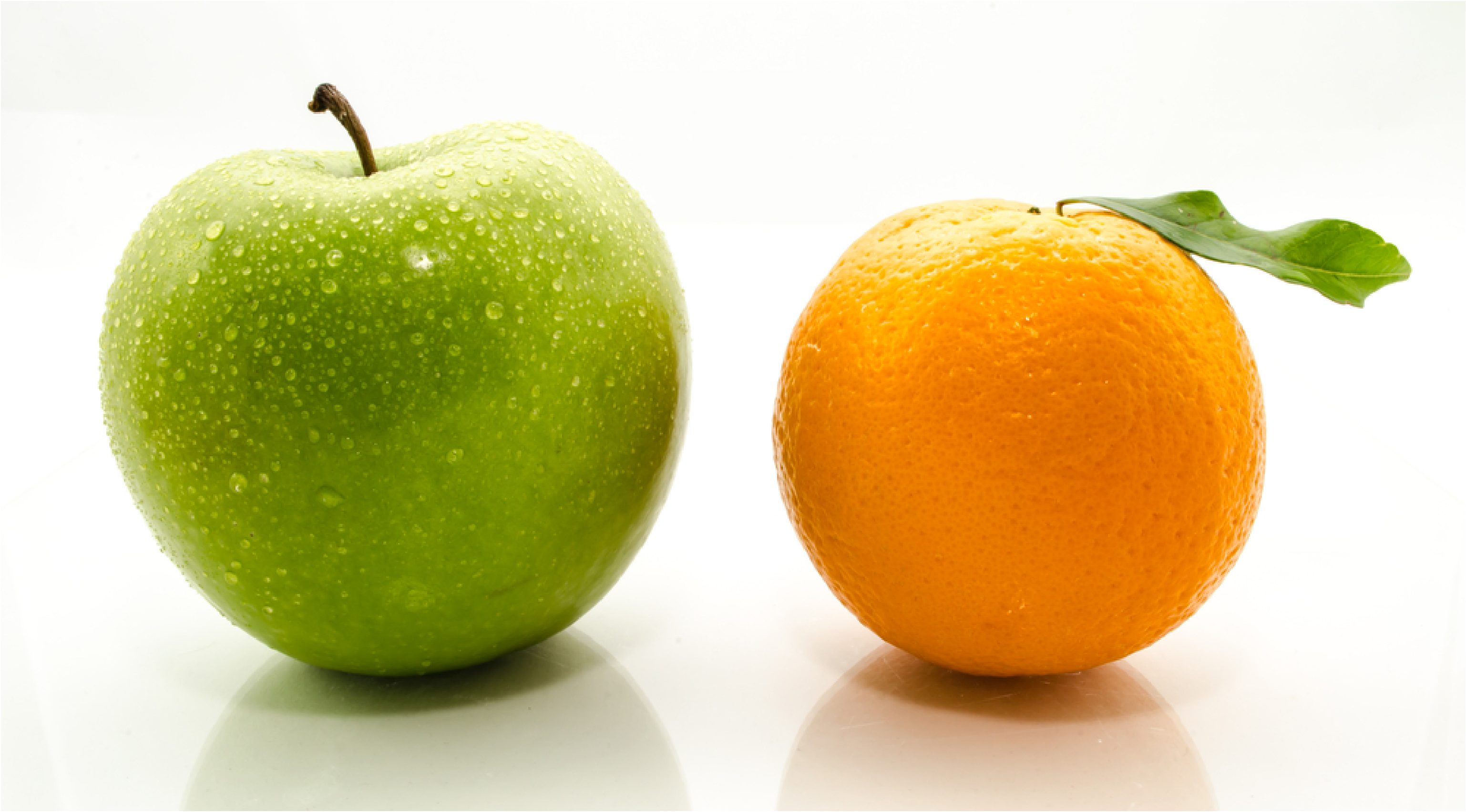
Values are bitwise XOR secret-shared

- Allows free XOR and evaluation of AND gates using MTs

Example: $x \stackrel{?}{=} y$



Part 4) Empirical Comparison



Setting

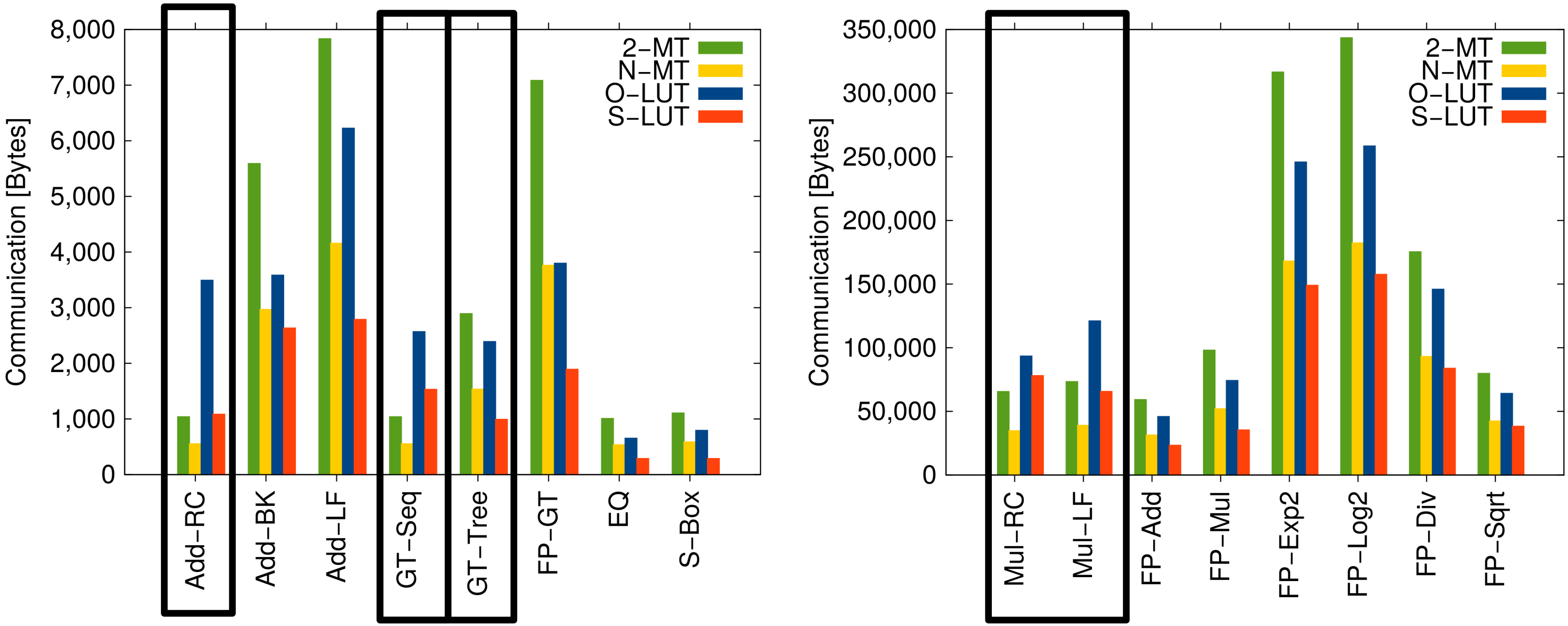
Evaluate communication and rounds for basic operations

- Addition
- Multiplication
- Comparison
- AES S-Box
- Floating-Point Operations

Evaluate different approaches

- 2-MT: GMW using 1oo2 OT extension (260 bits)
- N-MT: GMW using 1ooN OT extension (138 bits)
- O-LUT: for LUTs with up to 4 inputs
- S-LUT: for LUTs with up to 8 inputs

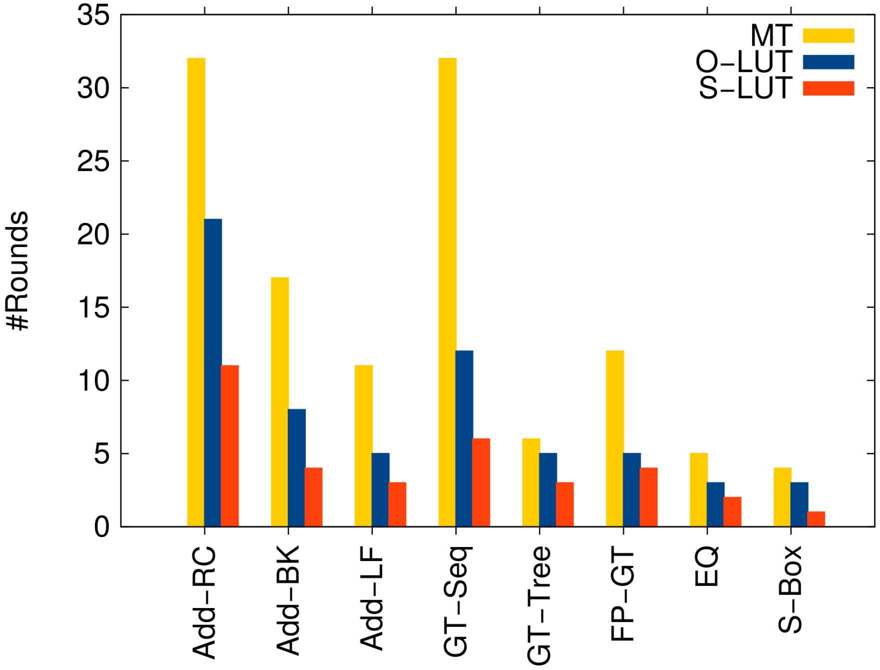
Communication Basic Operations

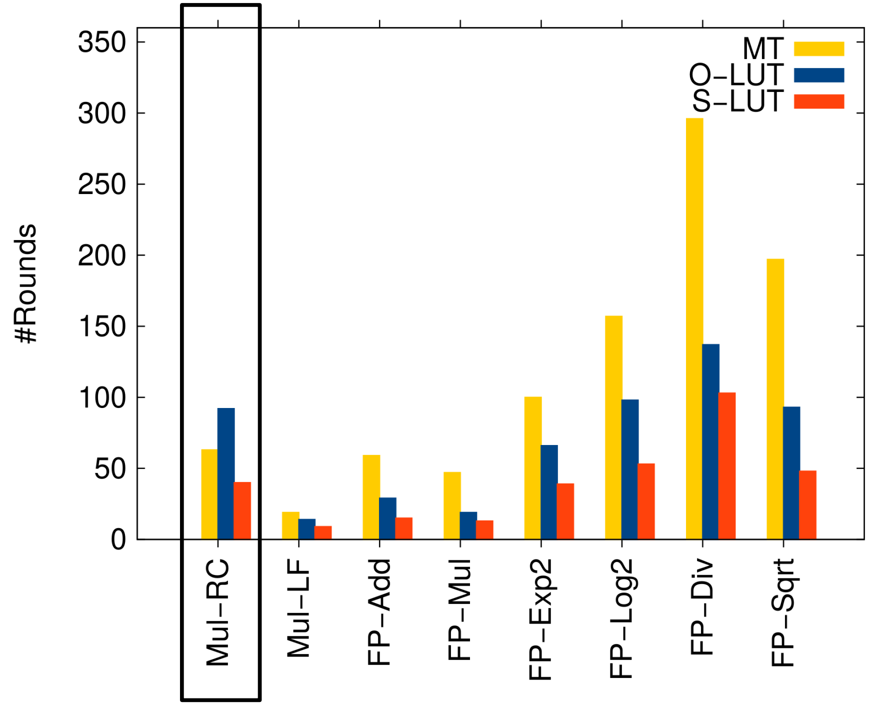


- Mostly: S-LUT < N-MT < O-LUT < 2-MT
- 2-MT and N-MT perform better for Ripple-carry based circuits
- LUT approaches perform best for tree based structures

EC SPRIDE

Rounds Basic Operations





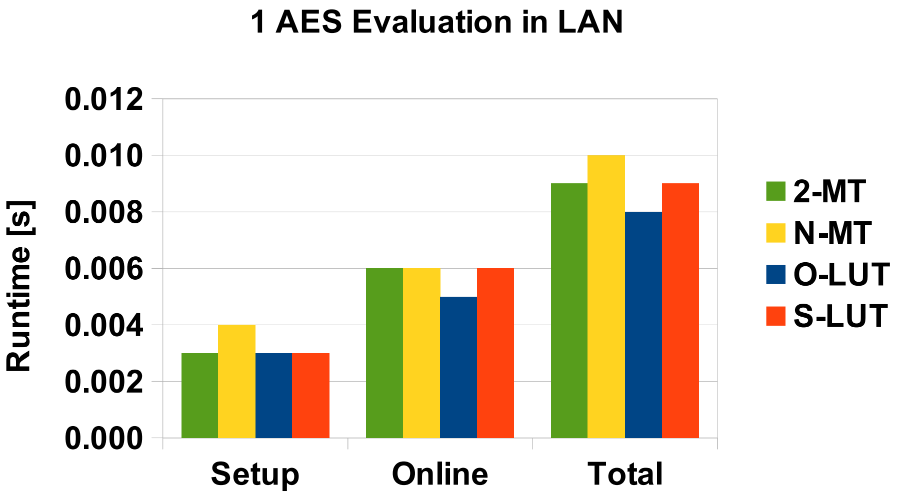
- Mostly: S-LUT < O-LUT < MT
- Exception: Multiplication with Ripple-carry addition

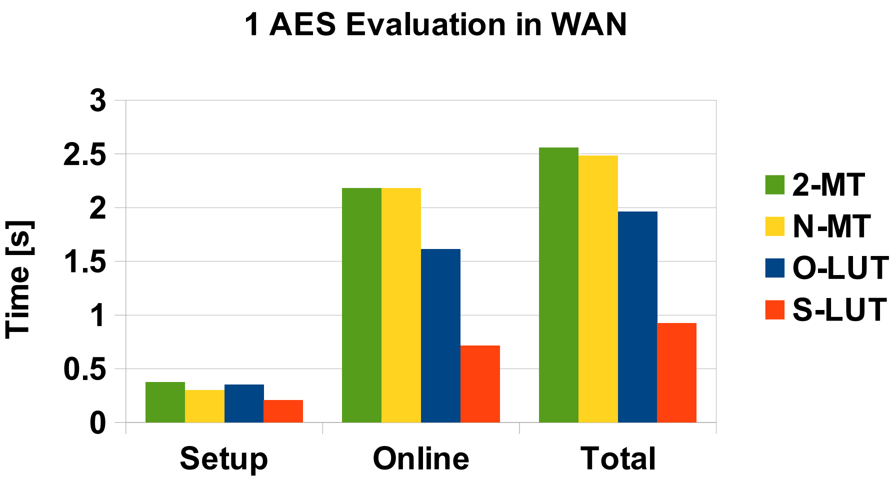
Evaluation on Applications: AES

TECHNISCHE UNIVERSITÄT DARMSTADT

AES encryption of 1 block using 4 threads

- LAN (1 GBit, 0.2 ms latency)
- WAN (120 Mbit, 100ms latency)



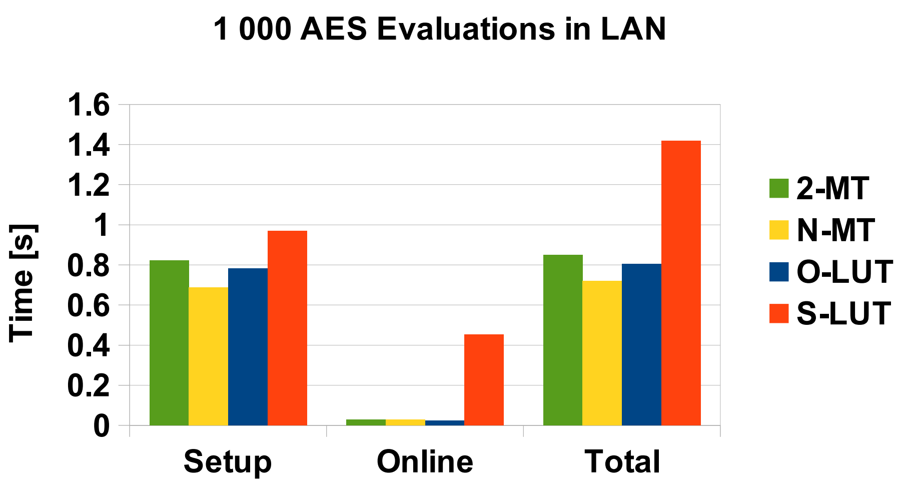


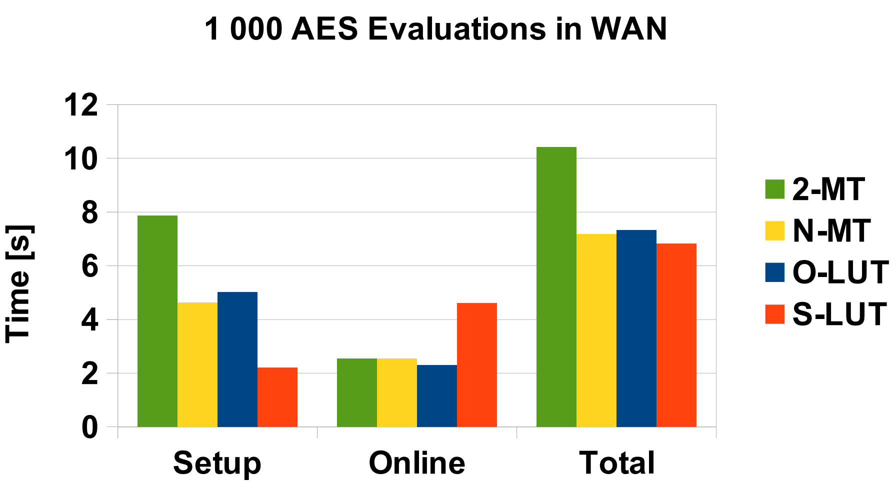
EC SPRIDE

Evaluation on Applications: AES

TECHNISCHE UNIVERSITÄT DARMSTADT

AES encryption of 1 000 blocks using 4 threads
- LAN (1 GBit, 0.2 ms latency)
- WAN (120 Mbit, 100ms latency)





EC SPRIDE

Take-Away Message

Traded more computation for less communication and rounds

GMW costs ~one ciphertext per AND

LUT protocols can reduce communication and rounds even further

Efficient OT Extension and its Impact on Secure Computation

TECHNISCHE UNIVERSITÄT DARMSTADT

Pushing the Communication Barrier of Passive Secure Two-Party Computation

Questions?

Contact: http://encrypto.de

EC SPRIDE

References

[BHKR13] M. Bellare, V. Hoang, S. Keelveedhi, P. Rogaway. Efficient Garbling from a fixed-key Cipher. In IEEE S&P 2013.

[DZ16] I. Damgård, R. W. Zakarias. Fast oblivious AES: A dedicated application of the MiniMac protocol. In Africacrypt 2016.

[GLNP15] S. Gueron, Y. Lindell, A. Nof, B. Pinkas. Fast garbling of circuits under standard assumptions. In CCS 2015.

[Huang12] Y. Huang. Practical secure two-party computation. Ph. D. Thesis 2012.

[IKMOP13] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, A. Paskin-Cherniasky. On the power of correlated randomness in secure computation. In TCC 2013.

[IKNP03] Y. Ishai, J. Kilian, K. Nissim, E. Petrank. Extending oblivious transfer efficiently. In CRYPTO 2013.

[KK12] V. Kolesnikov, R. Kumaresan. Improved secure two-party computation via information theoretic garbled circuits. In SCN 2012.

[KK13] V. Kolesnikov, R. Kumaresan. Improved OT extension for transferring short secrets. In CRYPTO 2013.

[KSS12] B. Kreuter, A. Shelat, C. Shen. Billion-gate secure computation with malicious adversaries. In USENIX 2012.

[ZRE15] S. Zahur, M. Rosulek, D. Evans. Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In EUROCRYPT 2015.