faster malicious 2pc with online/offline dual execution

Mike Rosulek @ Oregon State University OSU

collaborators:

Vladimir Kolesnikov @ Alcatel·Lucent

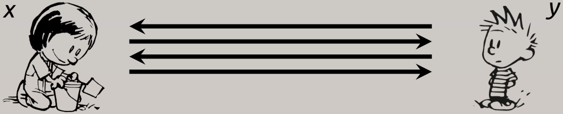
Payman Mohassel @ YAHOO!

Peter Rindal @ Oregon State University OSU

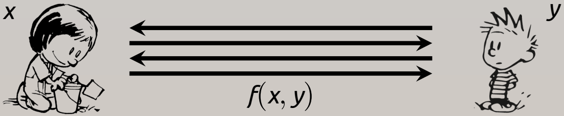
Ben Riva @ Bar-Ilan University אוניברסיטת בר-אילן

two-party secure computation

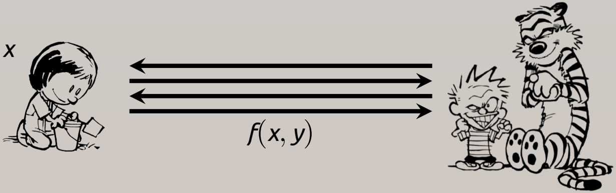
two-party secure computation



two-party secure computation



two-party secure computation



- Security against malicious adversaries

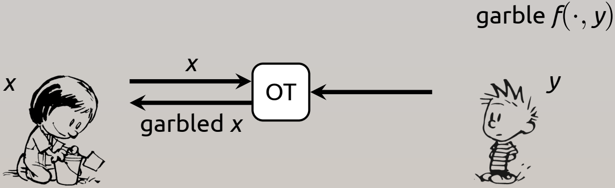
yao's 2pc protocol

x

garble $f(\cdot, y)$

y

yao’s 2pc protocol



garble $f(\cdot, y)$

x

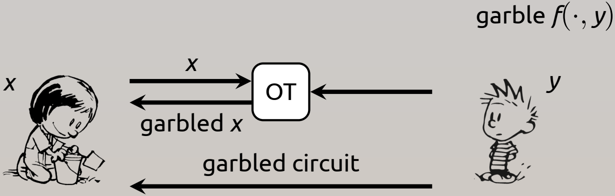
x

OT

garbled x

y

yao's 2pc protocol



garble $f(\cdot, y)$

x

x

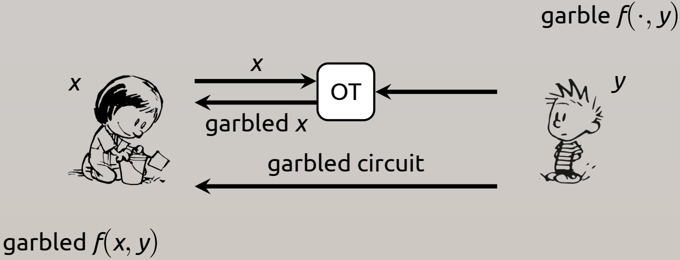
OT

garbled x

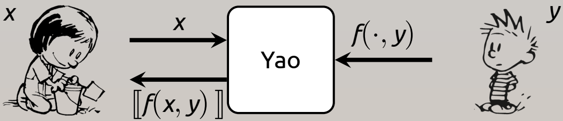
garbled circuit

y

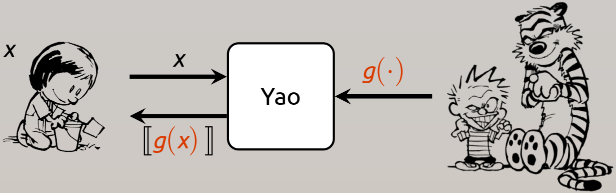
yao's 2pc protocol



yao's 2pc protocol

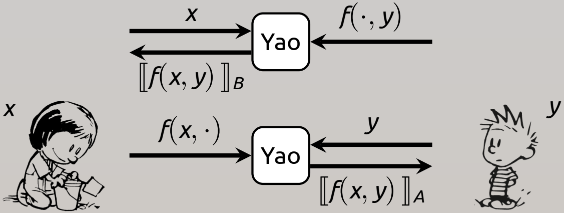


yao's 2pc protocol

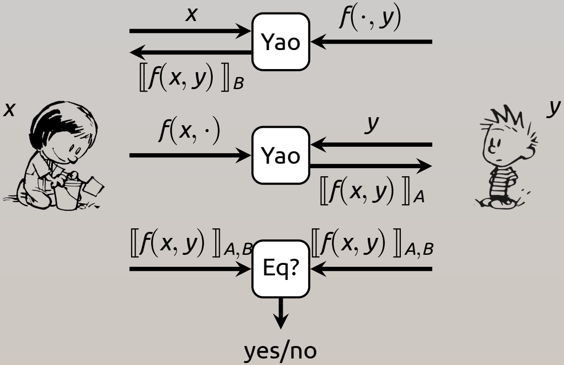


- Full security against malicious receiver
- Malicious sender can construct bad garbled circuit

dual execution protocol [MohasselFranklin06]

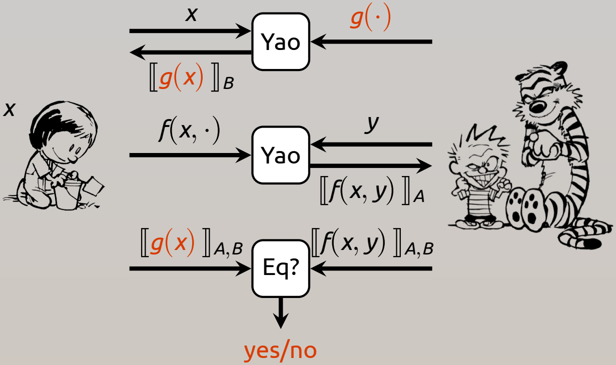


dual execution protocol [MohasselFranklin06]



- Define a **common** garbled encoding: $[\![z]\!]_{A,B} \overset{\text{def}}{=} [\![z]\!]_A \oplus [\![z]\!]_B$

dual execution protocol [MohasselFranklin06]



x

$g(\cdot)$

Yao

$\llbracket g(x) \rrbracket_B$

x

$f(x, \cdot)$

Yao

y

$\llbracket f(x, y) \rrbracket_A$

$\llbracket g(x) \rrbracket_{A,B}$

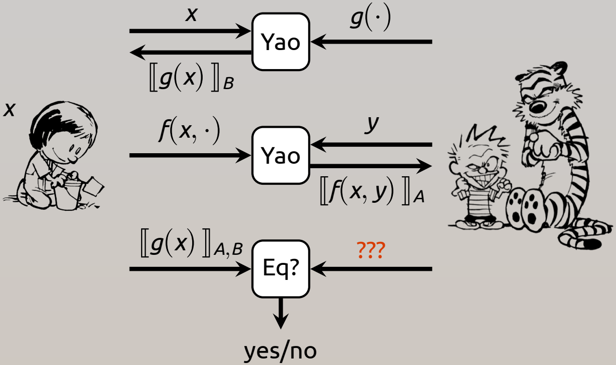
Eq?

$\llbracket f(x, y) \rrbracket_{A,B}$

yes/no

- Define a **common** garbled encoding: $\llbracket z \rrbracket_{A,B} \overset{\text{def}}{=} \llbracket z \rrbracket_A \oplus \llbracket z \rrbracket_B$
- Malicious Bob learns whether $g(x) = f(x, y)$ (only 1 bit)

dual execution protocol [MohasselFranklin06]

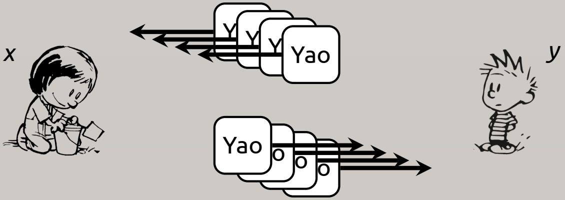


- Define a **common** garbled encoding: $[\![z]\!]_{A,B} \overset{\text{def}}{=} [\![z]\!]_A \oplus [\![z]\!]_B$
- Malicious Bob learns whether $g(x) = f(x, y)$ (only 1 bit)
- Malicious Bob can't predict $[\![z]\!]_{A,B}$ for for $z \neq f(x, y)$

  $\Rightarrow$ can't make Alice accept incorrect output!

reducing leakage [KolesnikovMohasselRivaR15]

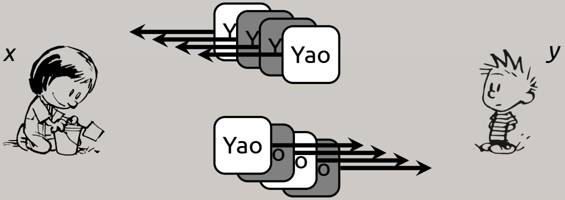
reducing leakage [KolesnikovMohasselRivaR15]



Main idea:

- Run κ copies of Yao's protocol in each direction

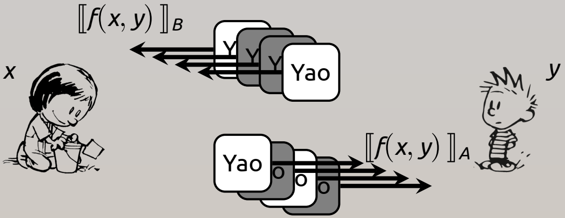
reducing leakage [KolesnikovMohasselRivaR15]



Main idea:

- Run κ copies of Yao's protocol in each direction
- Cut and choose: check each garbled circuit with probability 1/2.

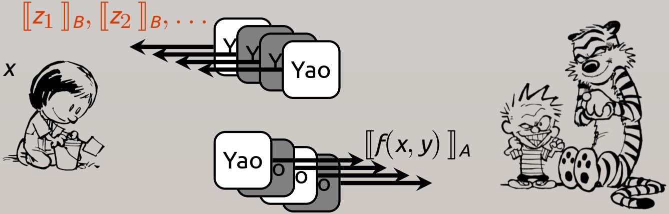
reducing leakage [KolesnikovMohasselRivaR15]



Main idea:

- Run κ copies of Yao's protocol in each direction
- Cut and choose: check each garbled circuit with probability 1/2.
- Garbled circuits in same direction have same output encoding

reducing leakage [KolesnikovMohasselRivaR15]



Main idea:

- Run κ copies of Yao's protocol in each direction
- Cut and choose: check each garbled circuit with probability 1/2.
- Garbled circuits in same direction have same output encoding
- What to do when Alice gets disagreeing outputs?

reconciliation technique

$[\![z^*]\!]_B$

$[\![z^*]\!]_A$

- Honest parties can compute common $[\![z^*]\!]_{A,B} \overset{\text{def}}{=} [\![z^*]\!]_B \oplus [\![z^*]\!]_A$

reconciliation technique

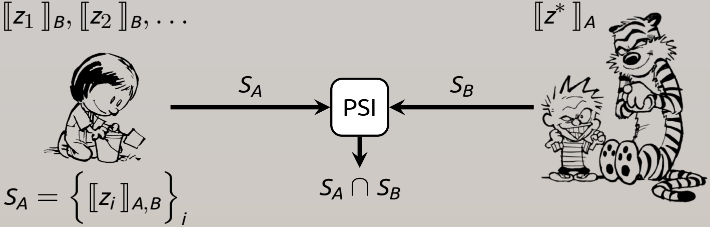
$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$[\![z^*]\!]_A$

- Honest parties can compute common $[\![z^*]\!]_{A,B} \overset{\text{def}}{=} [\![z^*]\!]_B \oplus [\![z^*]\!]_A$
- If disagreeing outputs, compute **set of candidates**

reconciliation technique



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

$[\![z^*]\!]_A$

S_A

PSI

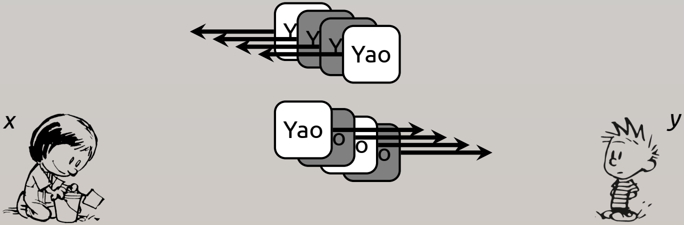
S_B

$S_A \cap S_B$

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

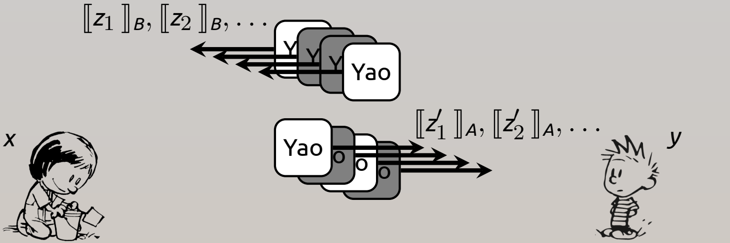
- Honest parties can compute common $[\![z^*]\!]_{A,B} \overset{\text{def}}{=} [\![z^*]\!]_B \oplus [\![z^*]\!]_A$
- If disagreeing outputs, compute **set of candidates**
- Do **private set intersection** on the sets!
  - $\Rightarrow$ PSI output identifies the “correct” z_i

protocol summary



- κ instances of Yao in each direction, check random subset

protocol summary

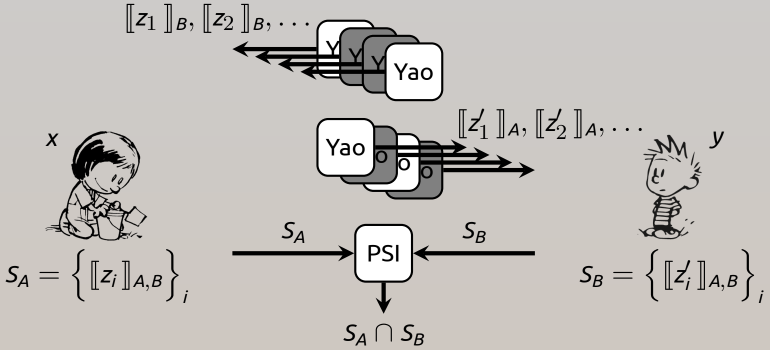


$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$
Y
Y
Y
Yao
x
Yao
o
o
o
$[\![z'_1]\!]_A, [\![z'_2]\!]_A, \ldots$
y

$$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i \qquad S_B = \left\{ [\![z'_i]\!]_{A,B} \right\}_i$$

- κ instances of Yao in each direction, check random subset
- Compute set of reconciliation values

protocol summary



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

Yao

$[\![z'_1]\!]_A, [\![z'_2]\!]_A, \ldots$

Yao

x

y

S_A

PSI

S_B

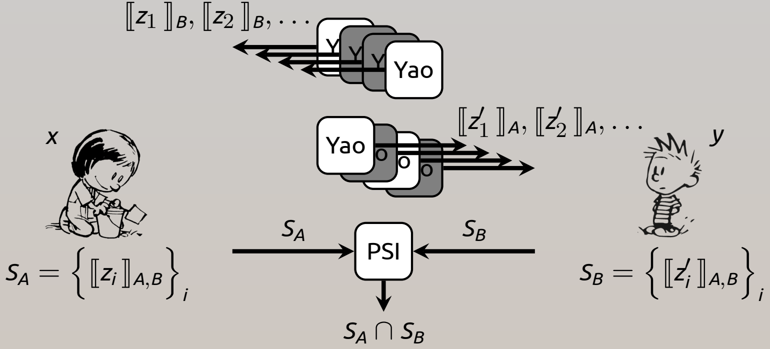
$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$S_B = \left\{ [\![z'_i]\!]_{A,B} \right\}_i$

$S_A \cap S_B$

- κ instances of Yao in each direction, check random subset
- Compute set of reconciliation values
- Private set intersection to identify correct output

protocol analysis



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

Yao

$[\![z'_1]\!]_A, [\![z'_2]\!]_A, \ldots$

Yao

x

y

S_A

PSI

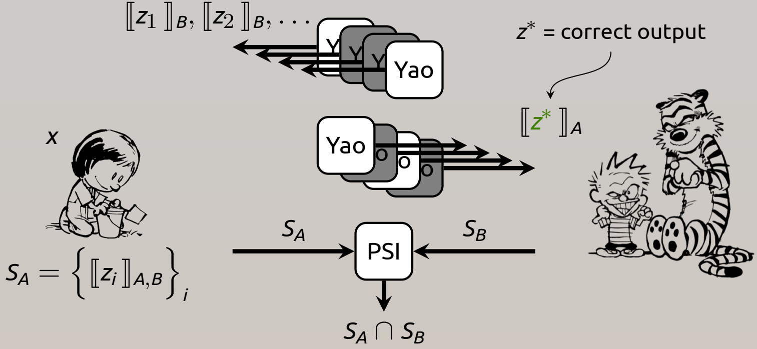
S_B

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$S_B = \left\{ [\![z'_i]\!]_{A,B} \right\}_i$

$S_A \cap S_B$

protocol analysis: corrupt Bob



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

Yao

z^* = correct output

$[\![z^*]\!]_A$

x

Yao

S_A

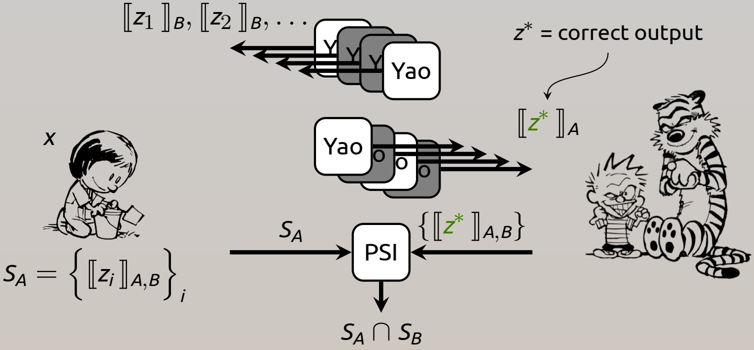
PSI

S_B

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$S_A \cap S_B$

protocol analysis: corrupt Bob



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

Yao

z^* = correct output

$[\![z^*]\!]_A$

x

Yao

S_A

PSI

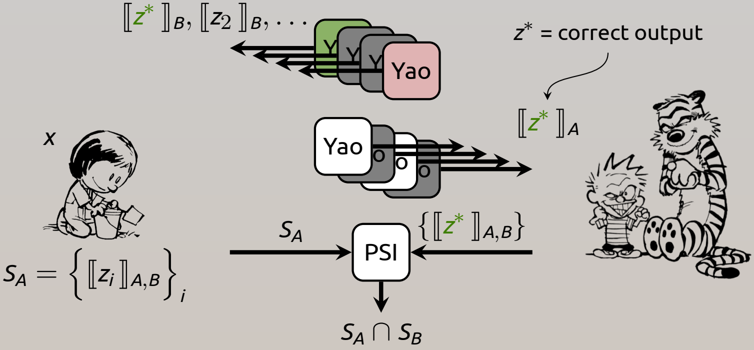
$\{[\![z^*]\!]_{A,B}\}$

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$S_A \cap S_B$

- Bob's only "useful" PSI input is $[\![z^*]\!]_{A,B}$

protocol analysis: corrupt Bob



$[\![z^*]\!]_B, [\![z_2]\!]_B, \ldots$

Yao

z^* = correct output

$[\![z^*]\!]_A$

x

Yao

S_A

PSI

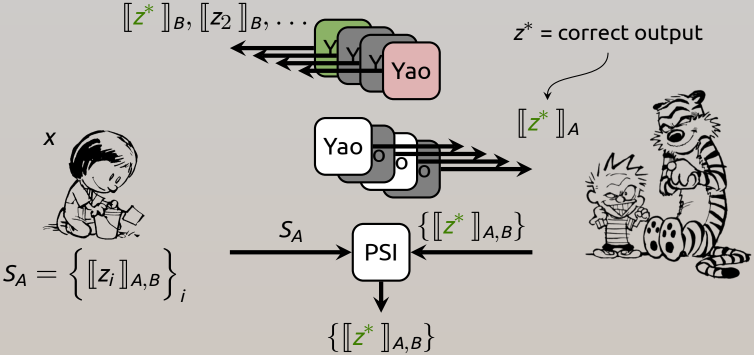
$\{[\![z^*]\!]_{A,B}\}$

$S_A \cap S_B$

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

- Bob's only "useful" PSI input is $[\![z^*]\!]_{A,B}$
- One of Bob's garbled circuits **correct** $\Rightarrow$

protocol analysis: corrupt Bob



$[\![z^*]\!]_B, [\![z_2]\!]_B, \ldots$

z^* = correct output

$[\![z^*]\!]_A$

x

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

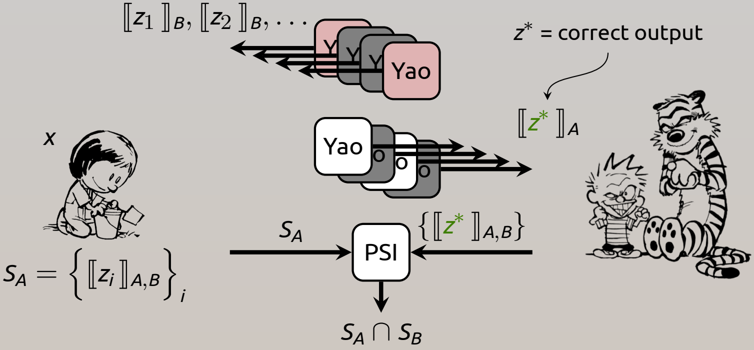
S_A

$\{[\![z^*]\!]_{A,B}\}$

$\{[\![z^*]\!]_{A,B}\}$

- Bob’s only “useful” PSI input is $[\![z^*]\!]_{A,B}$
- One of Bob’s garbled circuits **correct** $\Rightarrow$ PSI output leaks nothing!

protocol analysis: corrupt Bob



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

z^* = correct output

$[\![z^*]\!]_A$

x

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

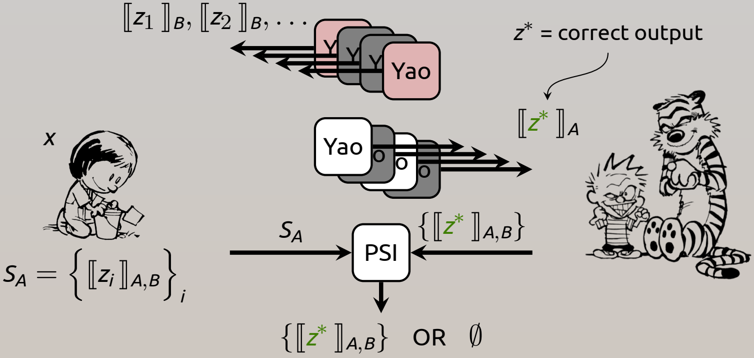
S_A

$\{[\![z^*]\!]_{A,B}\}$

$S_A \cap S_B$

- Bob's only "useful" PSI input is $[\![z^*]\!]_{A,B}$
- One of Bob's garbled circuits **correct** $\Rightarrow$ PSI output leaks nothing!
- None of Bob's garbled circuits correct $\Rightarrow$

protocol analysis: corrupt Bob



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

Y Y Y Yao

z^* = correct output

$[\![z^*]\!]_A$

x

Yao o o o

S_A

PSI

$\{[\![z^*]\!]_{A,B}\}$

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$\{[\![z^*]\!]_{A,B}\}$ OR $\emptyset$

- Bob's only "useful" PSI input is $[\![z^*]\!]_{A,B}$
- One of Bob's garbled circuits **correct** $\Rightarrow$ PSI output leaks nothing!
- None of Bob's garbled circuits correct $\Rightarrow$ PSI output leaks just 1 bit

“dual-ex+PSI” summary

κ garbled circuits in each direction (can be done simultaneously)

Adversary cannot violate output correctness

Adversary learns a single bit with probability $1/2^{\kappa}$; only when:

- All opened circuits are correct
- All evaluated circuits are incorrect

rest of the talk

Online/offline, multi-execution setting

- Reducing # of garbled circuits

Adapting “dual-execution+PSI” protocol to online/offline setting: [RindalR16]

- Ensuring input consistency
- Lightweight private set intersection

Implementation, performance

- Comparison to [LindellRiva15] and info-theoretic protocols

online/offline setting

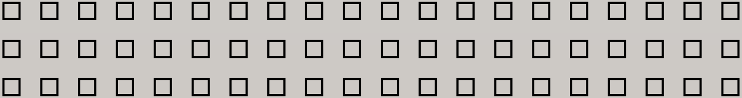
Want to do 2PC of same circuit N times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

online/offline setting

Want to do 2PC of same circuit *N* times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

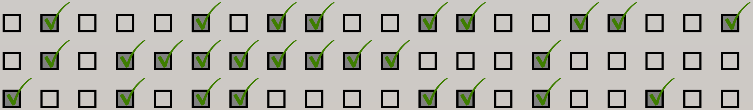


generate a lot of garbled circuits

online/offline setting

Want to do 2PC of same circuit *N* times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

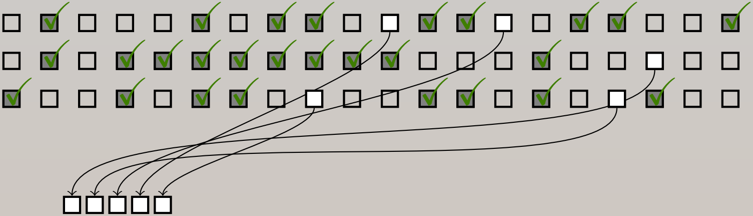


open and check some fraction of them

online/offline setting

Want to do 2PC of same circuit *N* times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

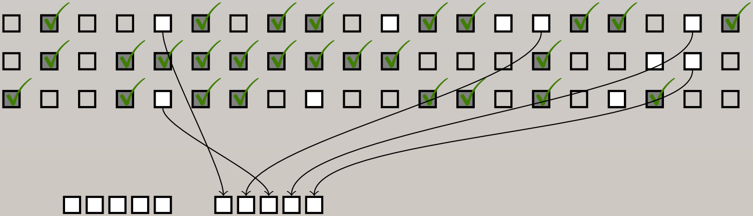


pick a random “bucket” of available circuits and evaluate them

online/offline setting

Want to do 2PC of same circuit *N* times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

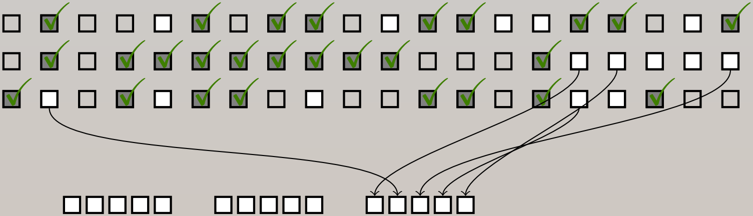


pick a random “bucket” of available circuits and evaluate them

online/offline setting

Want to do 2PC of same circuit *N* times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

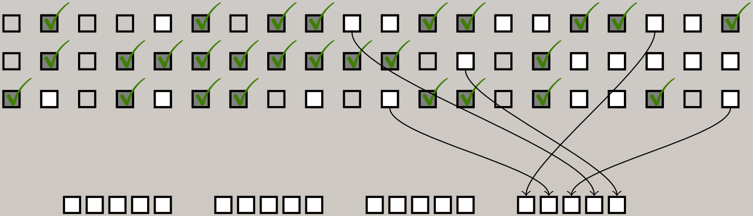


pick a random “bucket” of available circuits and evaluate them

online/offline setting

Want to do 2PC of same circuit *N* times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

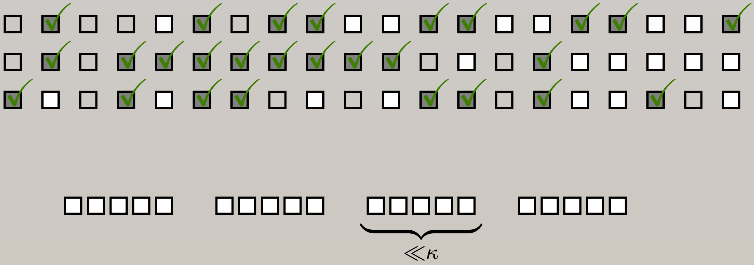


pick a random “bucket” of available circuits and evaluate them

online/offline setting

Want to do 2PC of same circuit N times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]



- for security $1/2^{\kappa}$, need $O(\kappa/\log N)$ circuits per execution
- example: $N = 1024$, $\kappa = 40 \implies$ 4 circuits per execution

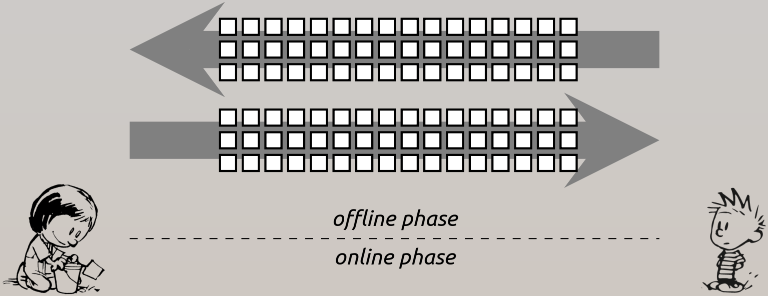
online/offline dual-ex [RindalR16]

offline phase

- -

online phase

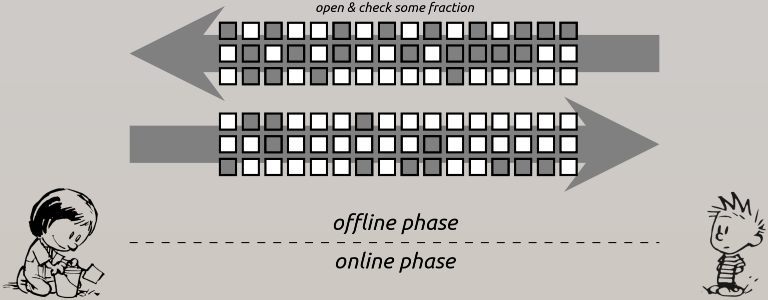
online/offline dual-ex [RindalR16]



offline phase

online phase

online/offline dual-ex [RindalR16]

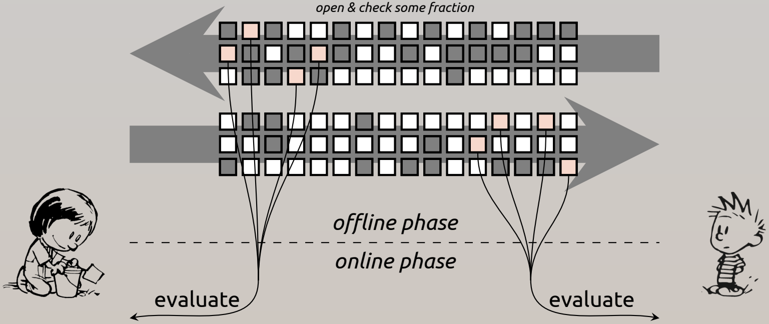


open & check some fraction

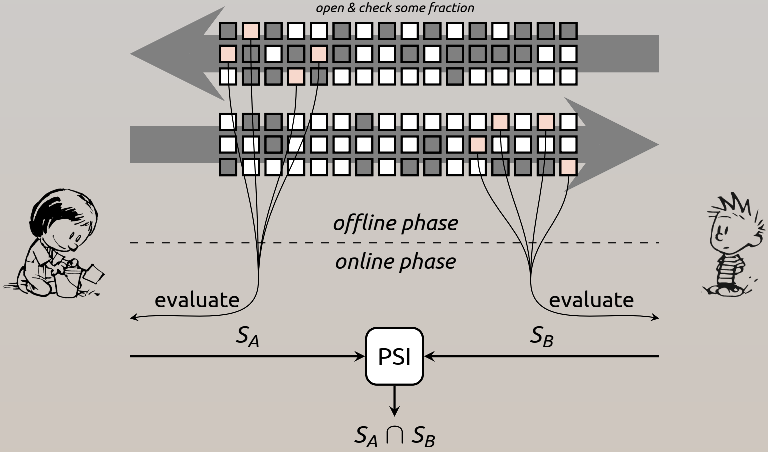
offline phase

online phase

online/offline dual-ex [RindalR16]



online/offline dual-ex [RindalR16]



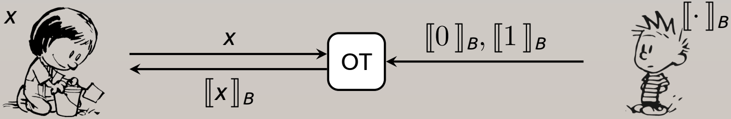
challenge #1: input consistency

How to ensure same inputs in Alice/Bob circuits?

[KolesnikovMohasselRivaR15,shelatShen13] technique incompatible with offline circuit garbling

pre-computing OTs [Beaver95]

[for simplicity: 1 input bit from Alice]



x

x

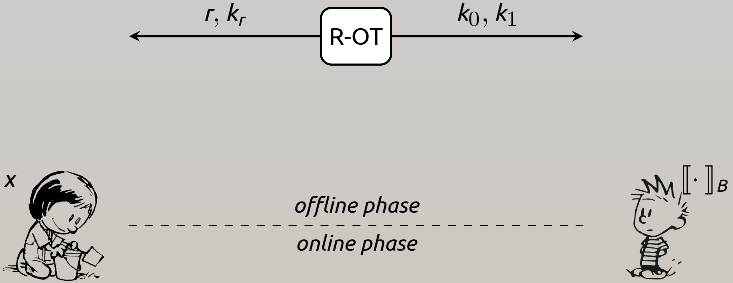
OT

$[\![x]\!]_B$

$[\![0]\!]_B, [\![1]\!]_B$

$[\![\cdot]\!]_B$

pre-computing OTs [Beaver95]



r, k_r

R-OT

k_0, k_1

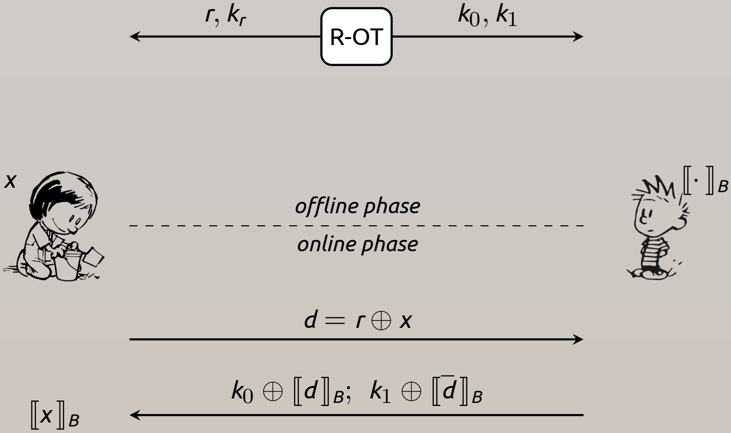
x

$[\![\cdot]\!]_B$

offline phase

online phase

pre-computing OTs [Beaver95]



r, k_r

R-OT

k_0, k_1

x

$[\![\cdot]\!]_B$

offline phase

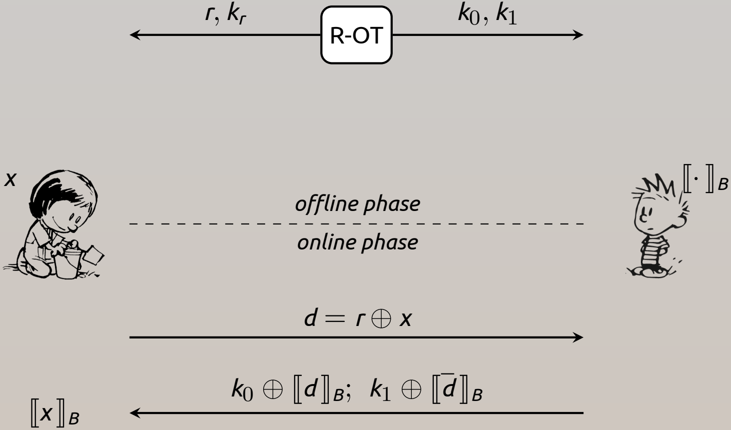
online phase

$d = r \oplus x$

$k_0 \oplus [\![d]\!]_B;\ \ k_1 \oplus [\![\bar{d}]\!]_B$

$[\![x]\!]_B$

consistency b/w A & B circuits



r, k_r

R-OT

k_0, k_1

x

$[\![\cdot]\!]_B$

offline phase

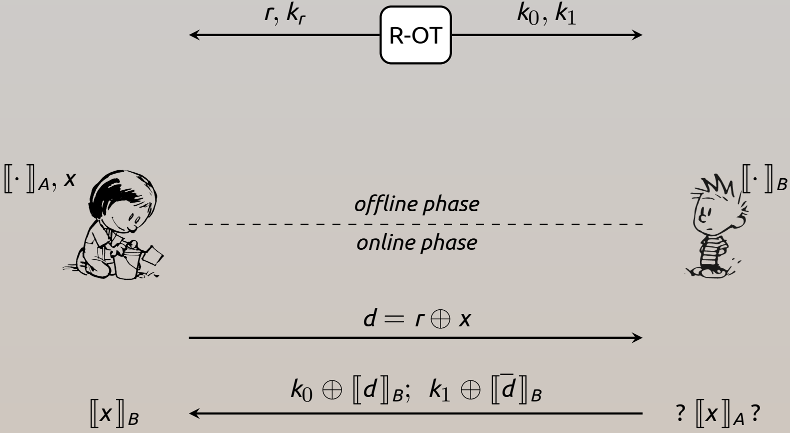
online phase

$d = r \oplus x$

$k_0 \oplus [\![d]\!]_B;\ \ k_1 \oplus [\![\bar{d}]\!]_B$

$[\![x]\!]_B$

consistency b/w A & B circuits



r, k_r R-OT k_0, k_1

$[\![\cdot]\!]_A, x$

$[\![\cdot]\!]_B$

offline phase

online phase

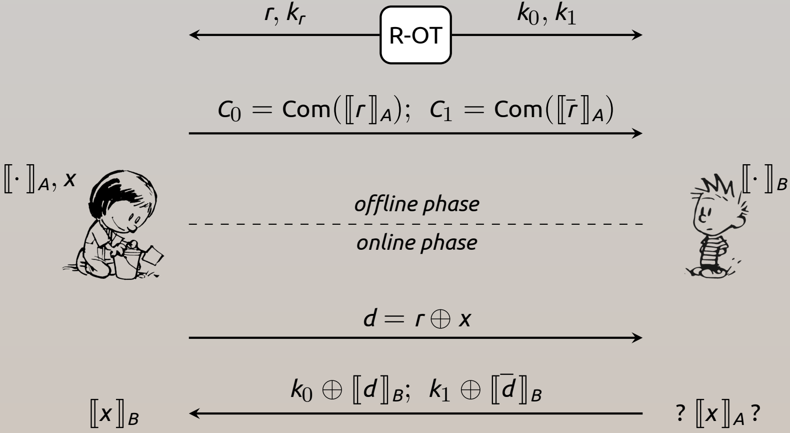
$d = r \oplus x$

$k_0 \oplus [\![d]\!]_B; \;\; k_1 \oplus [\![\bar{d}]\!]_B$

$[\![x]\!]_B$

? $[\![x]\!]_A$?

consistency b/w A & B circuits



r, k_r

R-OT

k_0, k_1

$C_0 = \mathsf{Com}(\llbracket r \rrbracket_A);\;\; C_1 = \mathsf{Com}(\llbracket \bar{r} \rrbracket_A)$

$\llbracket \cdot \rrbracket_A, x$

$\llbracket \cdot \rrbracket_B$

offline phase

online phase

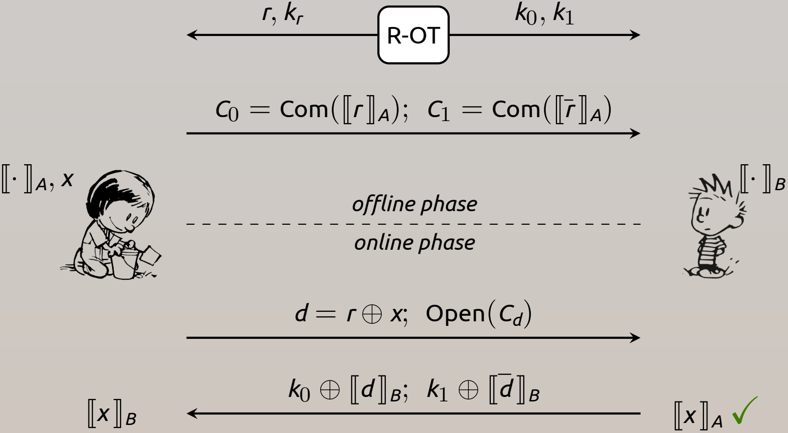
$d = r \oplus x$

$k_0 \oplus \llbracket d \rrbracket_B;\;\; k_1 \oplus \llbracket \bar{d} \rrbracket_B$

$\llbracket x \rrbracket_B$

? $\llbracket x \rrbracket_A$?

consistency b/w A & B circuits



r, k_r

R-OT

k_0, k_1

$C_0 = \mathrm{Com}(\llbracket r \rrbracket_A); \; C_1 = \mathrm{Com}(\llbracket \bar{r} \rrbracket_A)$

$\llbracket \cdot \rrbracket_A, x$

$\llbracket \cdot \rrbracket_B$

offline phase

online phase

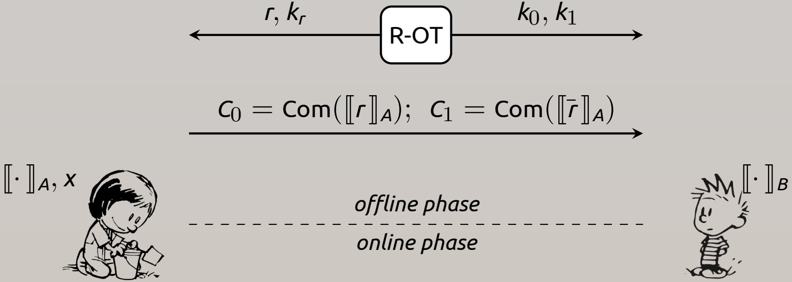
$d = r \oplus x; \; \mathrm{Open}(C_d)$

$k_0 \oplus \llbracket d \rrbracket_B; \; k_1 \oplus \llbracket \bar{d} \rrbracket_B$

$\llbracket x \rrbracket_B$

$\llbracket x \rrbracket_A$ ✓

consistency b/w A & B circuits



r, k_r

R-OT

k_0, k_1

$C_0 = \mathsf{Com}(\llbracket r \rrbracket_A); \;\; C_1 = \mathsf{Com}(\llbracket \bar{r} \rrbracket_A)$

$\llbracket \cdot \rrbracket_A, x$

$\llbracket \cdot \rrbracket_B$

offline phase

online phase

Can check consistency of commitments in cut-and-choose:

- Alice can show k_r to prove what r she got from OT

⇒ at least one pair of A/B circuits with consistency

input consistency

Within each bucket,

- Alice uses same input *x* on all Bob-circuits (easy)
- At least one Alice-circuit where Alice uses *x* (except prob $1/2^{\kappa}$)

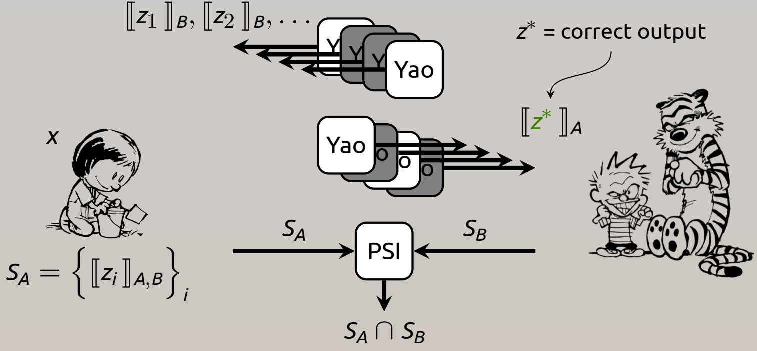
Suffices for security!

Zero online cost for input consistency!

challenge #2: psi

How to efficiently instantiate PSI?

closer look at PSI



$[\![z_1]\!]_B, [\![z_2]\!]_B, \dots$

Yao

z^* = correct output

$[\![z^*]\!]_A$

Yao

x

$S_A = \left\{[\![z_i]\!]_{A,B}\right\}_i$

S_A

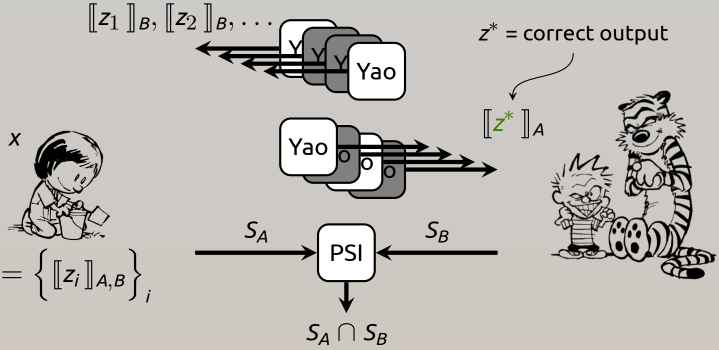
PSI

S_B

$S_A \cap S_B$

Bob’s only “useful” PSI input is $[\![z^*]\!]_{A,B}$

closer look at PSI



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

Yao

z^* = correct output

$[\![z^*]\!]_A$

x

Yao

S_A

PSI

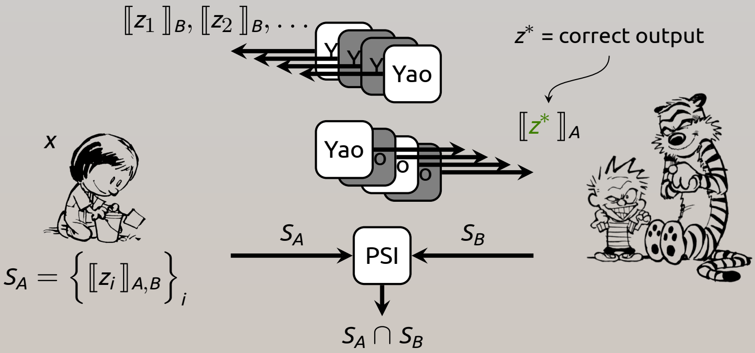
S_B

$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$

$S_A \cap S_B$

- Simulator knows $[\![z^*]\!]_{A,B}$; rest of S_A independent of Adv's view

closer look at PSI



$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$
Yao
z^* = correct output
$[\![z^*]\!]_A$
x
Yao
S_A
PSI
S_B
$S_A = \left\{ [\![z_i]\!]_{A,B} \right\}_i$
$S_A \cap S_B$

- Simulator knows $[\![z^*]\!]_{A,B}$; rest of S_A independent of Adv's view
- ⋆ Simulator **does not need to extract** Adv's input S_B!

  . . . it suffices to check whether $[\![z^*]\!]_{A,B} \in S_B$

instantiating psi

[KolesnikovMohasselRivaR15]:

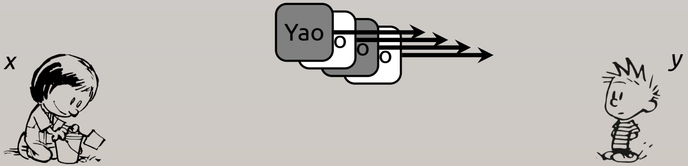
- Suggest using fully malicious PSI subprotocol

[RindalR16]:

- PSI protocol with “non-extracting security” suffices
- Implementation uses semi-honest PSI protocol of [PinkasSchneiderZohner14]
- Very cheap, based on pre-processed OTs (no public-key operations)

comparison to [LindellRiva15]

[LindellRiva14/15]: online/offline, malicious security, based on “traditional cut and choose” [Lindell13]:

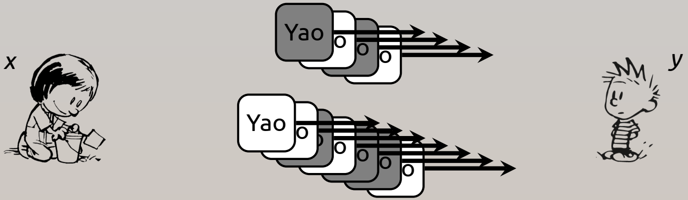


Two phases:

1. B circuits computing $f(x, y)$

comparison to [LindellRiva15]

[LindellRiva14/15]: online/offline, malicious security, based on "traditional cut and choose" [Lindell13]:

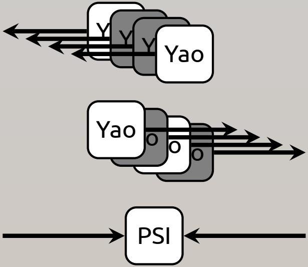


Two phases:

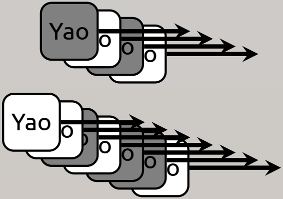
1. B circuits computing $f(x, y)$
2. $\sim 3B$ circuits computing:
   "if Bob can prove Alice cheated in phase 1, then reveal x to Bob"

protocol comparison

[KolesnikovMohasselRivaR15,RindalR16]:

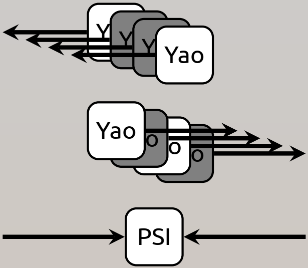


[LindellRiva14/15]:



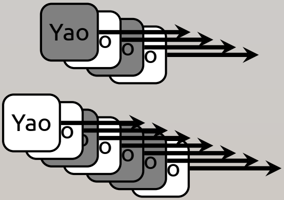
protocol comparison

[KolesnikovMohasselRivaR15,RindalR16]:



- *B* primary circuits in each direction (evaluated **simultaneously!**)

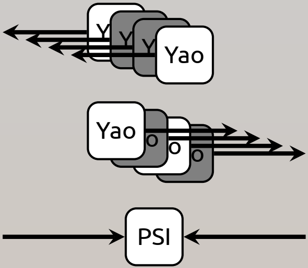
[LindellRiva14/15]:



- *B* primary circuits

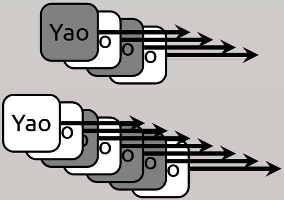
protocol comparison

[KolesnikovMohasselRivaR15,RindalR16]:



- *B* primary circuits in each direction (evaluated **simultaneously!**)
- PSI computation scales only with *B*

[LindellRiva14/15]:



- *B* primary circuits
- Aux computation scales as $3B \cdot \ell_{\text{input}}$

a closer look at κ

κ_c: Computational security parameter (e.g., 128)

κ_s: Statistical security parameter: security properties violated with probability $1/2^{\kappa_s}$ (e.g., 40)

"Traditional cut-and-choose" [LindellRiva14/15]:

- When cut-and-choose fails, adversary can completely break privacy & correctness

⇒ # of garbled circuits scales with κ_s

a closer look at κ

κ_c: Computational security parameter (e.g., 128)

κ_s: Statistical security parameter: security properties violated with probability $1/2^{\kappa_s}$ (e.g., 40)

⋆ κ_b: Bit-leaking parameter: privacy violated by a single bit (other security maintained) with probability $1/2^{\kappa_b}$

“Traditional cut-and-choose” [LindellRiva14/15]:

- When cut-and-choose fails, adversary can completely break privacy & correctness

$\Rightarrow$ # of garbled circuits scales with κ_s

a closer look at κ

κ_c: Computational security parameter (e.g., 128)

κ_s: Statistical security parameter: security properties violated with probability $1/2^{\kappa_s}$ (e.g., 40)

$\star$ κ_b: Bit-leaking parameter: privacy violated by a single bit (other security maintained) with probability $1/2^{\kappa_b}$

"Traditional cut-and-choose" [LindellRiva14/15]:

- When cut-and-choose fails, adversary can completely break privacy & correctness

$\Rightarrow$ # of garbled circuits scales with κ_s

Dual-execution approach [KolesnikovMohasselRivaR15,RindalR16]:

- When cut-and-choose fails, adversary learns only a bit

$\Rightarrow$ # of garbled circuits scales with $\kappa_b \leq \kappa_s$

some parameter possibilities

[RindalR16] with $\kappa_s = \kappa_b = 40$:

- same security as [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits

some parameter possibilities

[RindalR16] with $\kappa_s = \kappa_b = 40$:

- same security as [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits

[RindalR16] with $\kappa_s = 40$, $\kappa_b = 30$:

- same security as [LindellRiva] with $\kappa_s = 40$, except slightly higher probability of leaking single bit
- fewer garbled circuits (25% savings for $N = 1024$, 40% for $N = 512$)

some parameter possibilities

[RindalR16] with $\kappa_s = \kappa_b = 40$:

- same security as [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits

[RindalR16] with $\kappa_s = 40$, $\kappa_b = 30$:

- same security as [LindellRiva] with $\kappa_s = 40$, except slightly higher probability of leaking single bit
- fewer garbled circuits (25% savings for $N = 1024$, 40% for $N = 512$)

[RindalR16] with $\kappa_s = 80$, $\kappa_b = 40$:

- **strictly stronger** security than [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits (only PSI cost increases)

implementation

	[RindalR16]		[LindellRiva]		[DamgårdZakarias15]	
	offline	online	offline	online	offline	online
AES circuit	**5.1ms**	**1.3ms**	74ms	7ms	high?	6ms
SHA256 circuit	**48ms**	**8.4ms**	206ms	33ms	-	-

implementation

	[RindalR16]		[LindellRiva]		[DamgårdZakarias15]	
	offline	online	offline	online	offline	online
AES circuit	**5.1ms**	**1.3ms**	74ms	7ms	high?	6ms
SHA256 circuit	**48ms**	**8.4ms**	206ms	33ms	-	-

Amortized cost over $N = 1024$ executions:

- same hardware, LAN connection
  (Amazon `c4.8xlarge` = 36 core, 64GB RAM)
- same security ($\kappa_s = \kappa_b = 40$)

implementation

	[RindalR16]		[LindellRiva]		[DamgårdZakarias15]	
	offline	online	offline	online	offline	online
AES circuit	**5.1ms**	**1.3ms**	74ms	7ms	high?	6ms
SHA256 circuit	**48ms**	**8.4ms**	206ms	33ms	-	-

Amortized cost over $N = 1024$ executions:

- same hardware, LAN connection (Amazon `c4.8xlarge` = 36 core, 64GB RAM)
- same security ($\kappa_s = \kappa_b = 40$)

Maximum **throughput:** 0.26ms / AES block (3800+ Hz)

- [DamgårdZakarias15] reports 0.4ms

summary

Online-offline dual execution:

- Fastest 2PC with malicious security to date: 1.3ms AES
- Some protocol advantages over "classic" cut-and-choose

Future work:

summary

Online-offline dual execution:

- Fastest 2PC with malicious security to date: 1.3ms AES
- Some protocol advantages over “classic” cut-and-choose

Future work:

	garbled circ	info-theoretic
online latency	low ✓ ✓	low ✓
online throughput	high ✓	high ✓ ✓
constant rounds?	yes ✓	no ✗
offline time	low ✓	(very) high ✗
function-indep pre-processing?	no ✗	yes ✓

summary

Online-offline dual execution:

- Fastest 2PC with malicious security to date: 1.3ms AES
- Some protocol advantages over “classic” cut-and-choose

Future work: Combine GC with info-theoretic? (at least for 2-party)

	garbled circ	info-theoretic	hybrid
online latency	low ✓ ✓	low ✓	low ✓
online throughput	high ✓	high ✓ ✓	high ✓ ✓
constant rounds?	yes ✓	no ✗	no ✗
offline time	low ✓	(very) high ✗	low ✓
function-indep pre-processing?	no ✗	yes ✓	yes ✓

the end; thanks.

Richer Efficiency/Security Tradeoffs in 2PC

Vladimir Kolesnikov, Payman Mohassel, Ben Riva & Mike Rosulek

`ia.cr/2015/055`

Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution

Peter Rindal & Mike Rosulek

in submission