richer efficiency/security tradeoffs in 2PC

Vladimir Kolesnikov @ Alcatel·Lucent
Payman Mohassel @ YAHOO!
Ben Riva @ Bar-Ilan University אוניברסיטת בר-אילן
Mike Rosulek @ Oregon State University OSU

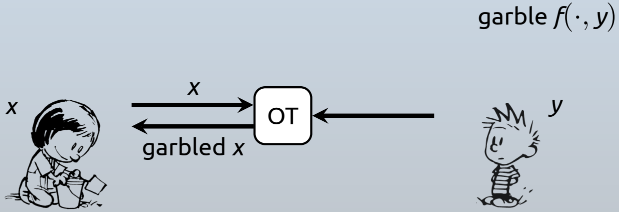
yao's 2pc protocol

garble $f(\cdot, y)$

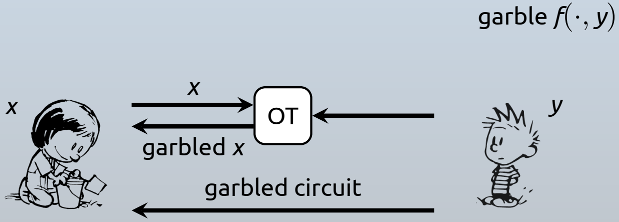
x

y

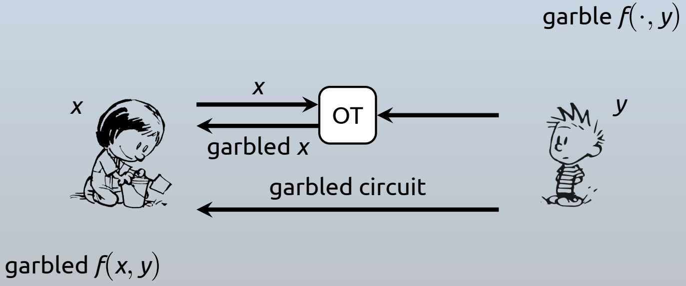
yao’s 2pc protocol



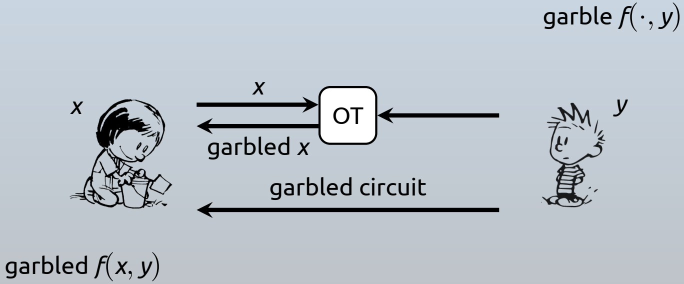
yao’s 2pc protocol



yao's 2pc protocol

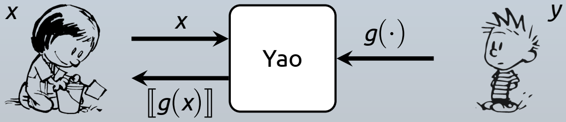


yao's 2pc protocol



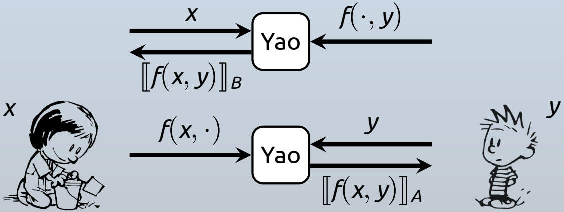
- Secure against semi-honest sender & malicious receiver
- Malicious sender can construct bad garbled circuit

yao's 2pc protocol

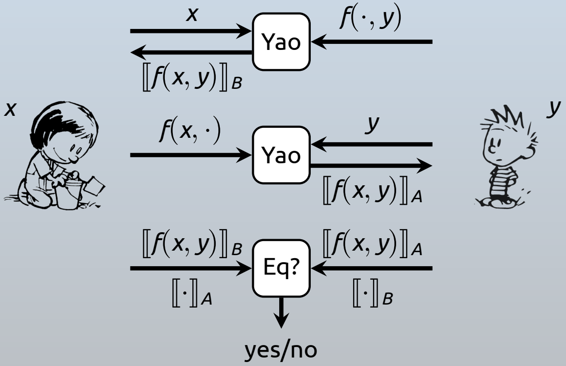


- Secure against semi-honest sender & malicious receiver
- Malicious sender can construct bad garbled circuit

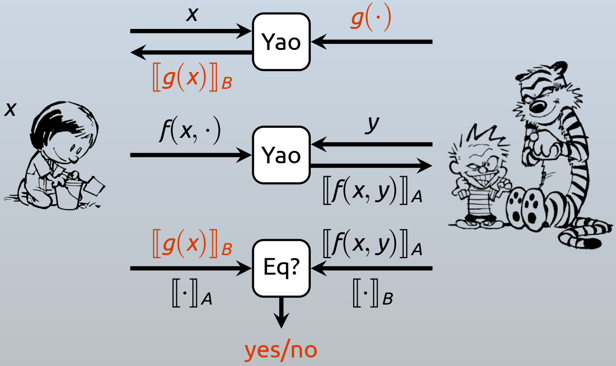
dual execution protocol [MohasselFranklin06]



dual execution protocol [MohasselFranklin06]



dual execution protocol [MohasselFranklin06]



- Malicious Bob learns whether $g(x) \stackrel{?}{=} f(x, y)$ for arbitrary g
- That's *all* he learns (i.e., only 1 bit) [MohasselFranklin06]
- Correctness never violated: Alice never accepts a wrong output.

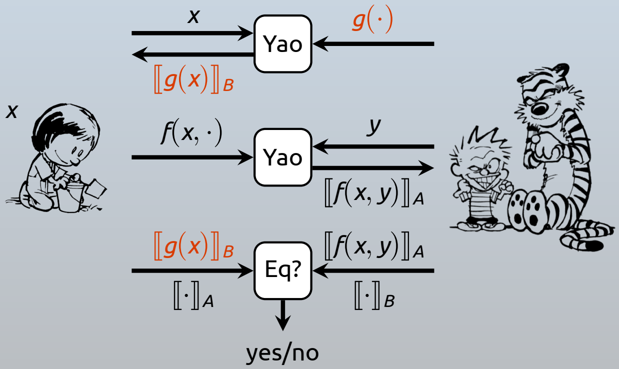
what's your paper about?

Improvements to the dual-execution mechanism:

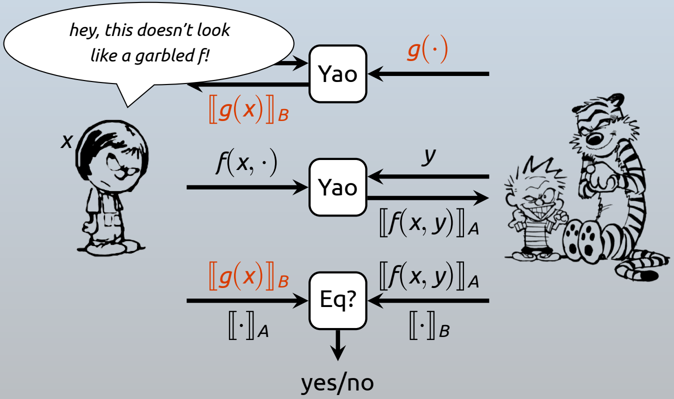
1. Restrict nature of the leaked bit
2. Reduce probability of a bit leaking

restrict nature of leaked bit

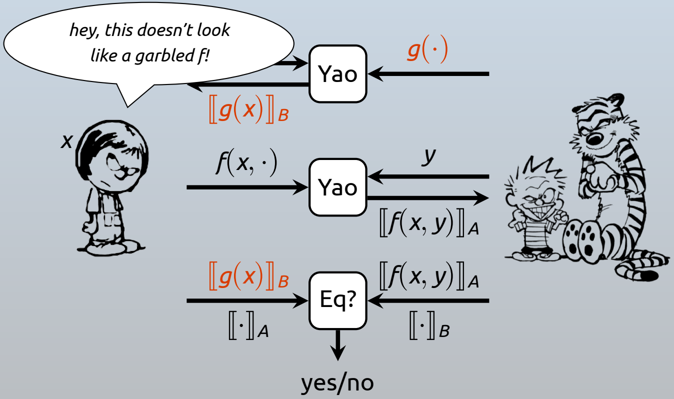
natural way to restrict leakage



natural way to restrict leakage



natural way to restrict leakage



- "Sanity checking" garbled circuit (# gates, topology, etc) should restrict leakage [HuangKatzEvans12]

topology-enforcing garbling

Folklore: In a standard garbling scheme, you can at least infer the *topology* of a malicious garbled circuit.

topology-enforcing garbling

Folklore: In a standard garbling scheme, you can at least infer the *topology* of a malicious garbled circuit.

Definition

Garbling scheme is **topology-enforcing** if

$\exists$ extractor *Ext*

$\forall$ (possibly malicious) garbled circuits F, garbled encoding info e

: $Ext(F, e) \to$ plain circuit f:

- f "explains" output behavior of F
- (apparent) topology of F = topology of f

(can also define "property-enforcing" for arbitrary properties)

topology-enforcing garbling

Folklore: In a standard garbling scheme, you can at least infer the *topology* of a malicious garbled circuit. Yes and no!

Definition

Garbling scheme is **topology-enforcing** if

$\exists$ extractor *Ext*

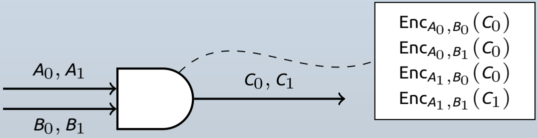
$\forall$ (possibly malicious) garbled circuits F, garbled encoding info e

: $Ext(F, e) \rightarrow$ plain circuit f:

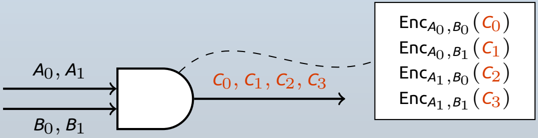
- f "explains" output behavior of F
- (apparent) topology of F = topology of f

(can also define "property-enforcing" for arbitrary properties)

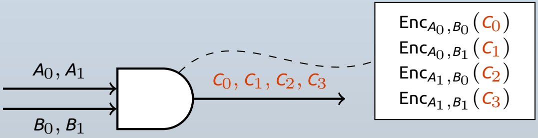
topology-enforcing garbling



topology-enforcing garbling

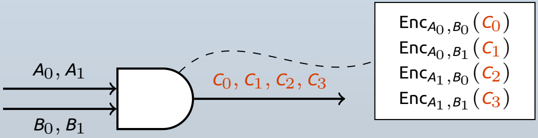


topology-enforcing garbling



- Standard schemes enforce topology but not *information bandwidth*
- Garbled circuit with one output wire can leak *entire input*.
- Achieve topology-enforcing (in ROM) by adding 2 hashes to each wire

topology-enforcing garbling



- Standard schemes enforce topology but not *information bandwidth*
- Garbled circuit with one output wire can leak *entire input*.
- Achieve topology-enforcing (in ROM) by adding 2 hashes to each wire

Theorem

Dual-execution protocol with topo-enforcing garbling scheme leaks:

$$g(x) \stackrel{?}{=} f(x, y)$$

for adversarially chosen g with **same topology** as $f(\cdot, y)$

further restricting the leakage

Dual execution checks that malicious circuit agrees with honest circuit...

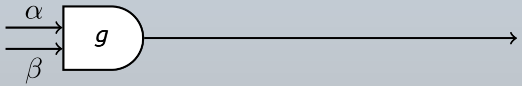
- in the **output wires only!**

further restricting the leakage

Dual execution checks that malicious circuit agrees with honest circuit...

- in the **output wires only!**

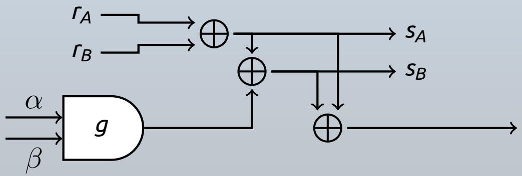
for every wire in circuit:



further restricting the leakage

Dual execution checks that malicious circuit agrees with honest circuit...

- in the **output wires only!**

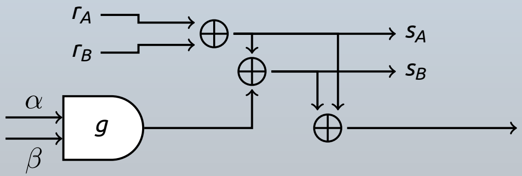


- Secret-share each wire value, output shares s_A, s_B, recombine shares
- Dual execution mechanism compares shares s_A, s_B against a *correct* circuit!

further restricting the leakage

Dual execution checks that malicious circuit agrees with honest circuit...

- in the **output wires only!**



- Secret-share each wire value, output shares s_A, s_B, recombine shares
- Dual execution mechanism compares shares s_A, s_B against a *correct* circuit!
- Now malicious circuit must agree with honest circuit on **all internal wires**

only computation leaks [MicaliReyzin04]

*"Cannot leak **jointly** on a and b unless they are computed on **simultaneously** at some step."*

only computation leaks [MicaliReyzin04]

*“Cannot leak **jointly** on a and b unless they are computed on **simultaneously** at some step.”*

Definition

OCL predicate in a circuit depends only on the *inputs to one single gate.*

only computation leaks [MicaliReyzin04]

*"Cannot leak **jointly** on a and b unless they are computed on **simultaneously** at some step."*

Definition

OCL predicate in a circuit depends only on the *inputs to one single gate.*

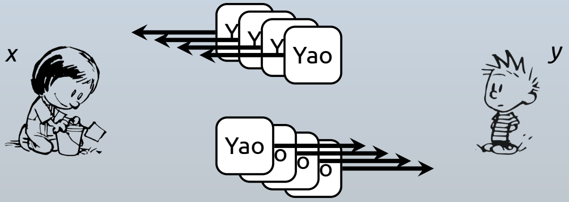
Theorem

- Transform circuit C to C^* using wire-secret-sharing construction
- Run dual-execution of C^* with topology-enforcing garbling scheme

$\Rightarrow$ Adversary learns only a **conjunction of OCL predicates** in C

reducing probability of leakage

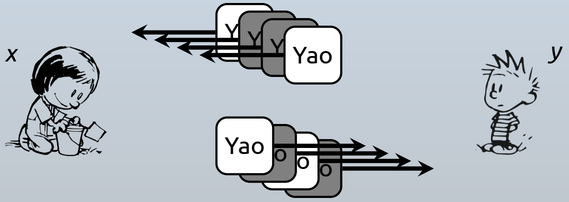
reducing probability of leakage



Main idea:

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

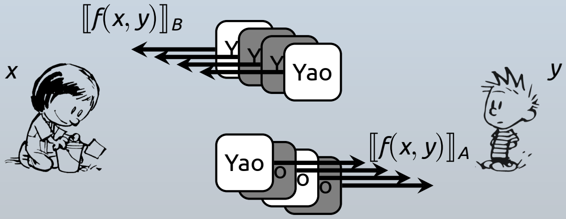
reducing probability of leakage



Main idea:

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

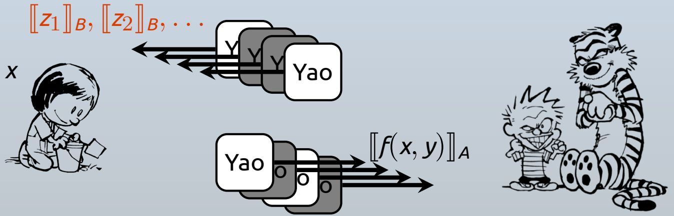
reducing probability of leakage



Main idea:

- Run *s* 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 probability of leakage



Main idea:

- Run s 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

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

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

- Two honest parties can compute common value $[\![z^*]\!]_B \oplus [\![z^*]\!]_A$

reconciliation technique

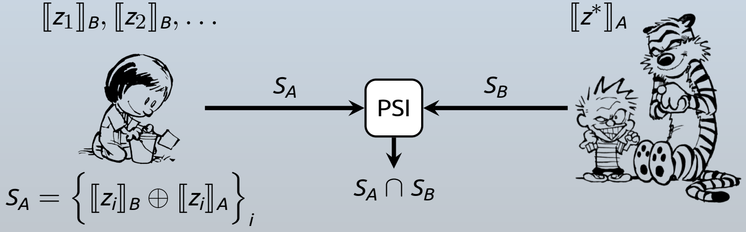
$[\![z_1]\!]_B, [\![z_2]\!]_B, \ldots$

$$S_A = \left\{ [\![z_i]\!]_B \oplus [\![z_i]\!]_A \right\}_i$$

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

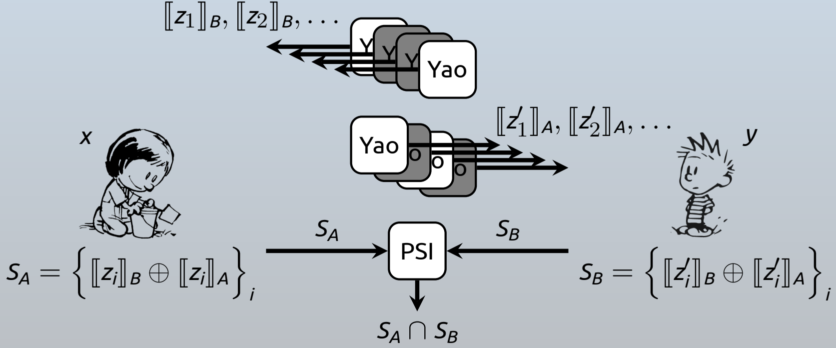
- Two honest parties can compute common value $[\![z^*]\!]_B \oplus [\![z^*]\!]_A$
- If disagreeing output, compute **set of candidates**

reconciliation technique

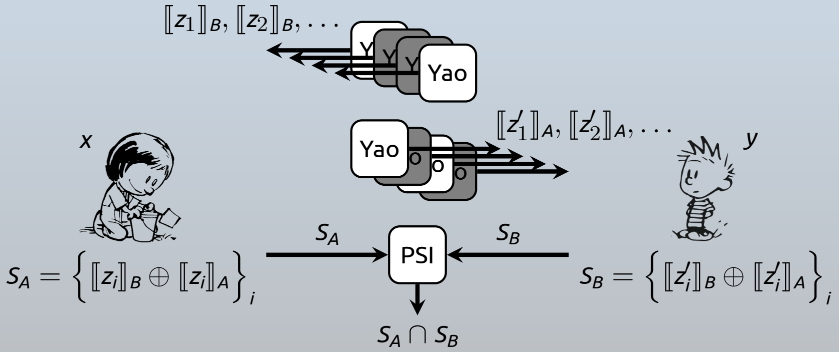


- Two honest parties can compute common value $[\![z^*]\!]_B \oplus [\![z^*]\!]_A$
- If disagreeing output, compute **set of candidates**
- Perform **private set intersection** on the sets!
  - Bob learns nothing from PSI unless **all** circuits evaluated by Alice are bad.

protocol summary



protocol summary



Output of PSI leaks extra information only if:

- **All** checked circuits are **good**, **all** evaluated circuits are **bad**

$\Rightarrow$ leakage with probability 2^{-s}

other details

Adversary cannot violate **correctness**, only privacy

- Privacy violated only by one bit, and only with probability 2^{-s}
- ϵ-CovIDA security notion of [MohasselRiva14]
- Compelling generalization of covert security [AumannLindell10]; **useful for smaller** s

other details

Adversary cannot violate **correctness**, only privacy

- Privacy violated only by one bit, and only with probability 2^{-s}
- ϵ-CovIDA security notion of [MohasselRiva14]
- Compelling generalization of covert security [AumannLindell10]; **useful for smaller** s

Alice & Bob can choose different "s" parameters!

other details

Adversary cannot violate **correctness**, only privacy

- Privacy violated only by one bit, and only with probability 2^{-s}
- ϵ-CovIDA security notion of [MohasselRiva14]
- Compelling generalization of covert security [AumannLindell10]; **useful for smaller** s

Alice & Bob can choose different “s” parameters!

Ensure same input used in all circuits?

- Compute $f(x, y) \| H_1(x) \| H_2(y)$ for universal hash H_i [shelatShen13]

other details

Adversary cannot violate **correctness**, only privacy

- Privacy violated only by one bit, and only with probability 2^{-s}
- ϵ-CovIDA security notion of [MohasselRiva14]
- Compelling generalization of covert security [AumannLindell10]; **useful for smaller** s

Alice & Bob can choose different "s" parameters!

Ensure same input used in all circuits?

- Compute $f(x, y) \| H_1(x) \| H_2(y)$ for universal hash H_i [shelatShen13]

Circuits have common output wire labels?

- Enforced via hash-commitments to output wire labels
- Commit to PSI input before opening circuits for cut-and-choose

comparison with other protocols

[MohasselRiva13]+[Lindell13]: 2^s security from *s* circuits:

- Receiver extracts the input of a cheating sender, computes *f* himself
- PSI significantly cheaper than input-recovery bootstrap circuit
- With probability 2^{-s}, adversary can violate all security properties

[HuanKatzEvans13]: 2^s security from *s* circuits each direction:

- Similar dual-execution setup, different (slower) reconciliation phase
- With probability 2^{-s}, adversary can learn more than 1 bit.
- Both parties must use same *s*.

summary

Restricting leakage predicate in dual-execution:

- Restrict to “Only Computation Leaks”-style leakage
- Formalize guarantees given by malicious garbled circuits

Reducing leakage probability in dual-execution:

- Cut and choose, reconcile using Private Set Intersection

koniec!

dziękuję!

Richer Efficiency/Security Tradeoffs in 2PC

Vladimir Kolesnikov, Payman Mohassel, Ben Riva & Mike Rosulek

`eprint.iacr.org/2015/055`