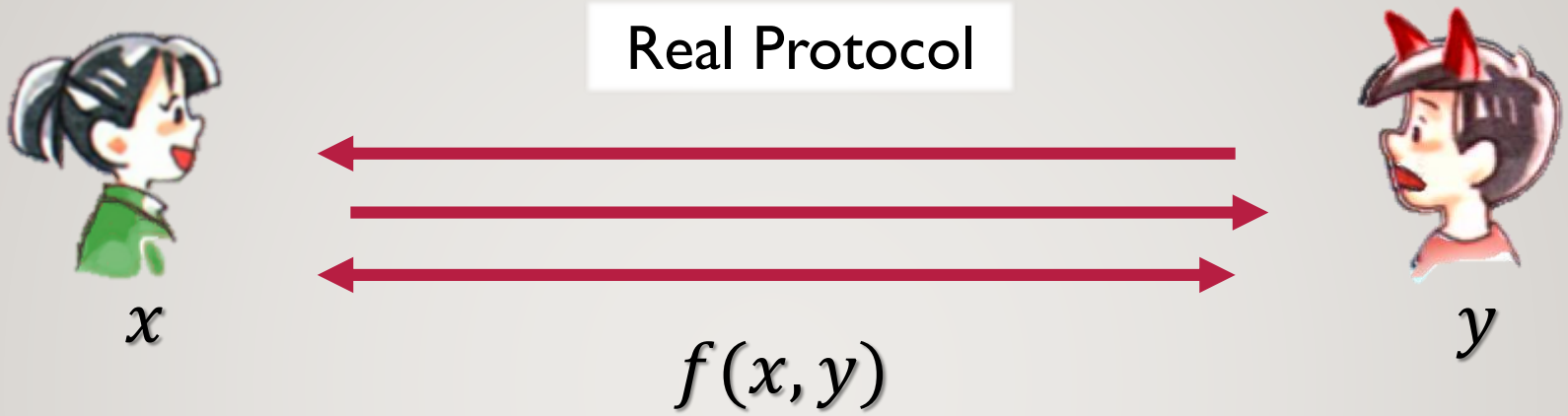
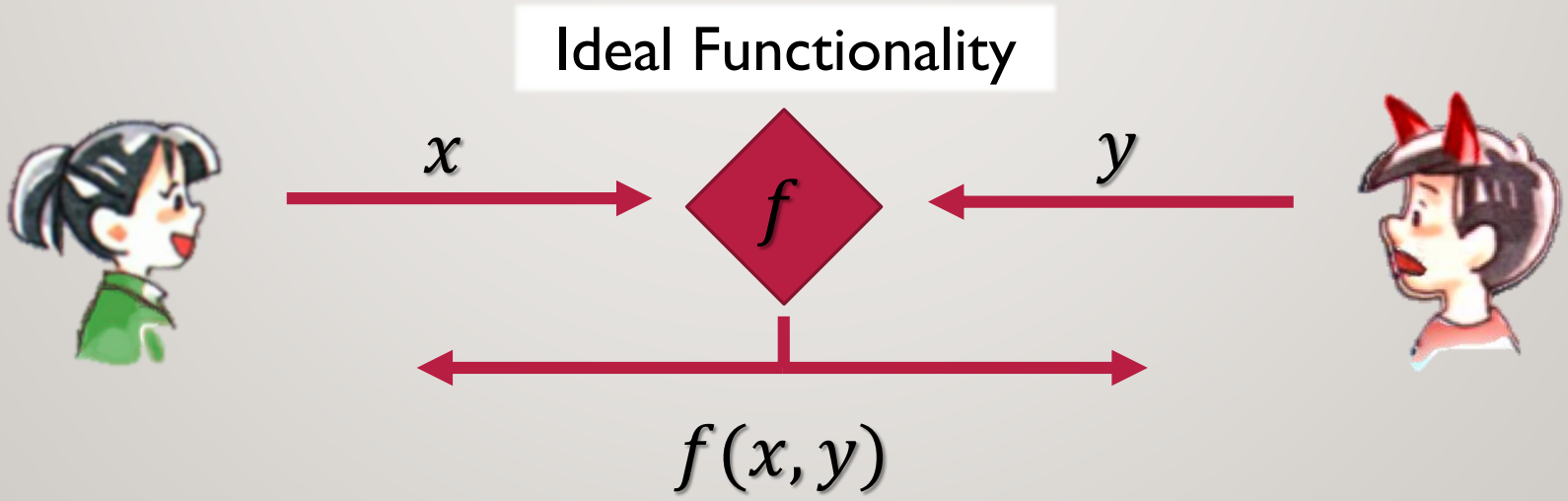
Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution

Peter Rindal
Mike Rosulek

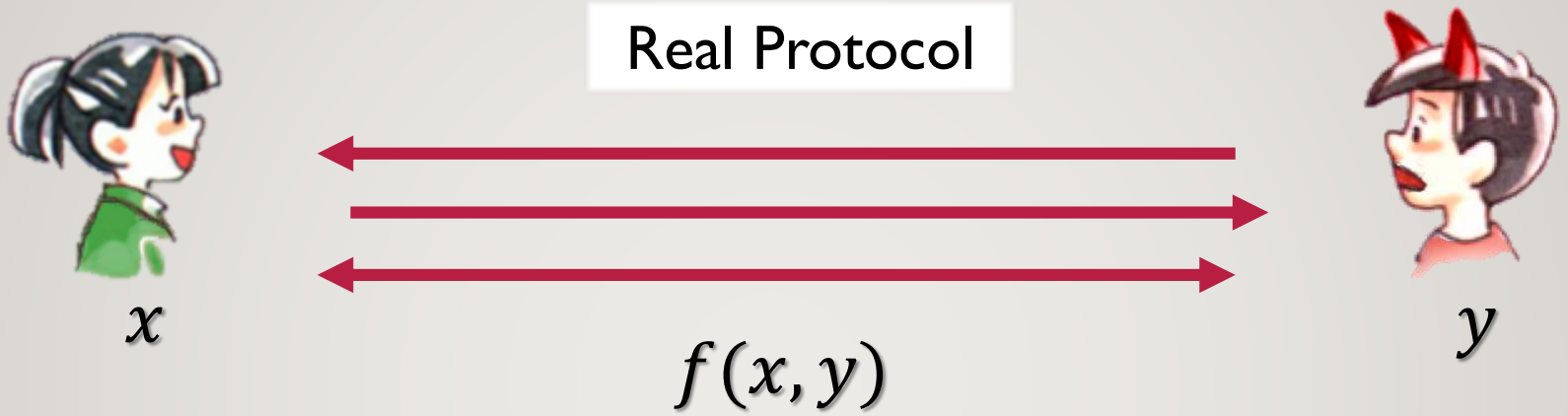
Oregon State University OSU

2 Party Computation





2 Party Computation



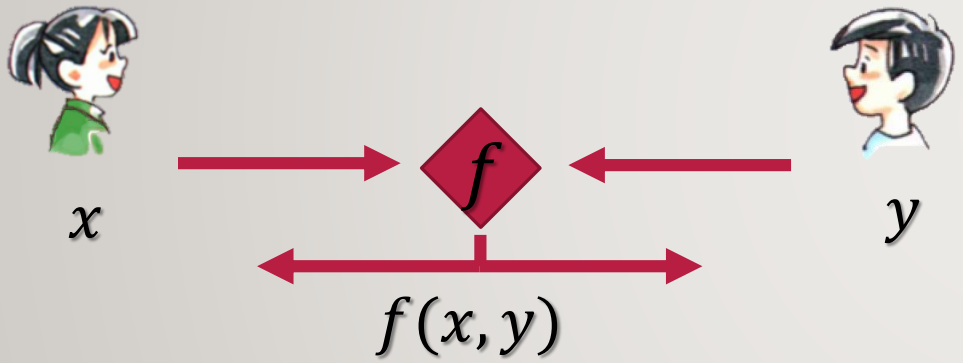
- Secure against malicious adversaries
- Def (simplified):

$$\forall \; \text{  } \; \exists \; \boxed{\mathcal{S}} :$$

$$\text{Real}_{\pi}(f,x,y) \approx \boxed{\mathcal{S}}(f,y,f(x,y))$$

Applications

2-party Secure Computation



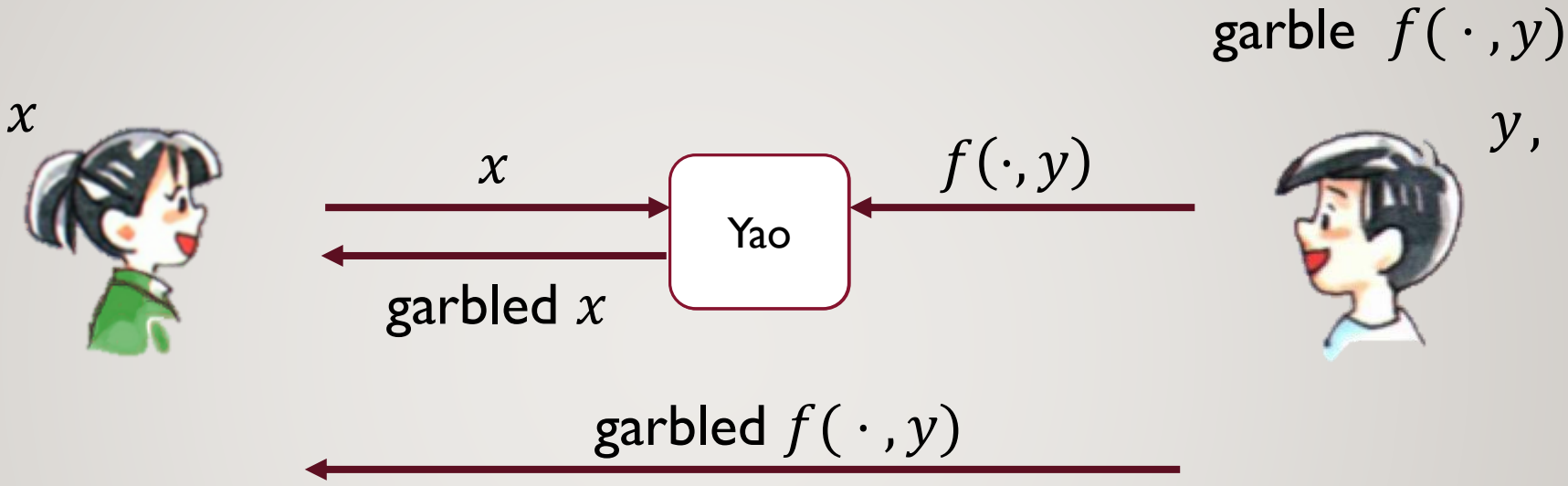
Applications

- Private database querying
  - Database x,
  - Query y

- Joint machine learning
  - Datasets x, y,
  - Model $= f(x,y)$

- Secure auctions
  - Bids x, y
  - Winning bid $= f(x,y)$

Yao's Protocol

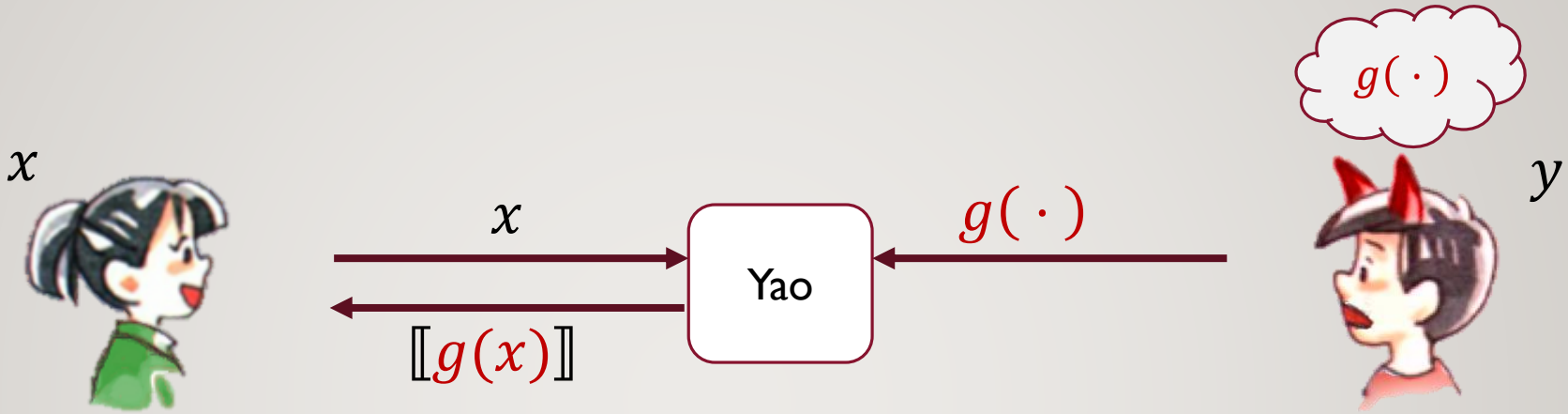


$[\![f(x,y)]\!] \stackrel{\text{def}}{=}$ garbled $f(x,y)$

Bob knows $[\![\cdot]\!]$

- Security properties:
  - Privacy – Alice learn no more than $f(x,y)$
  - Authenticity – Alice can not guess any output encoding other than $[\![f(x,y)]\!]$

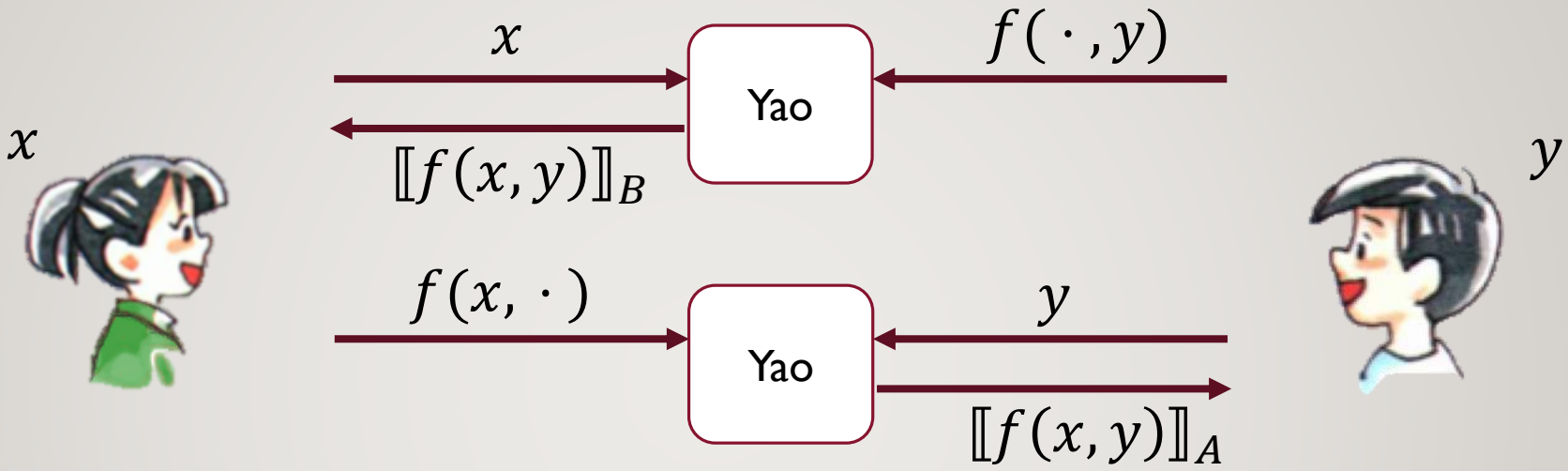
Yao's Protocol



Problems with malicious Adversaries

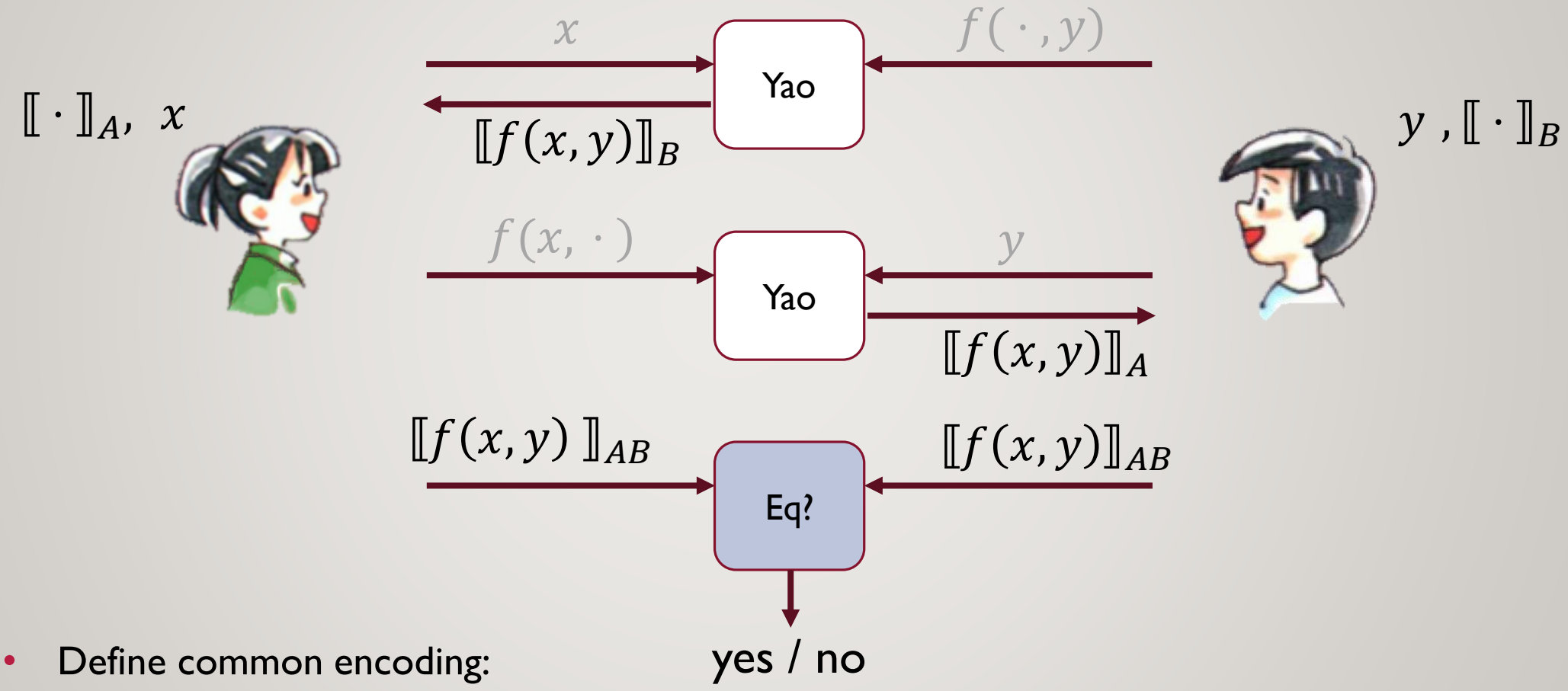
- The circuit may not be correctly constructed
  - E.g. $g(x) := x$
- May violate privacy and correctness
- Not always detectable

Dual Execution [MohasselFranklin06]



- First Yao secure against Alice.
- Second Yao secure against Bob

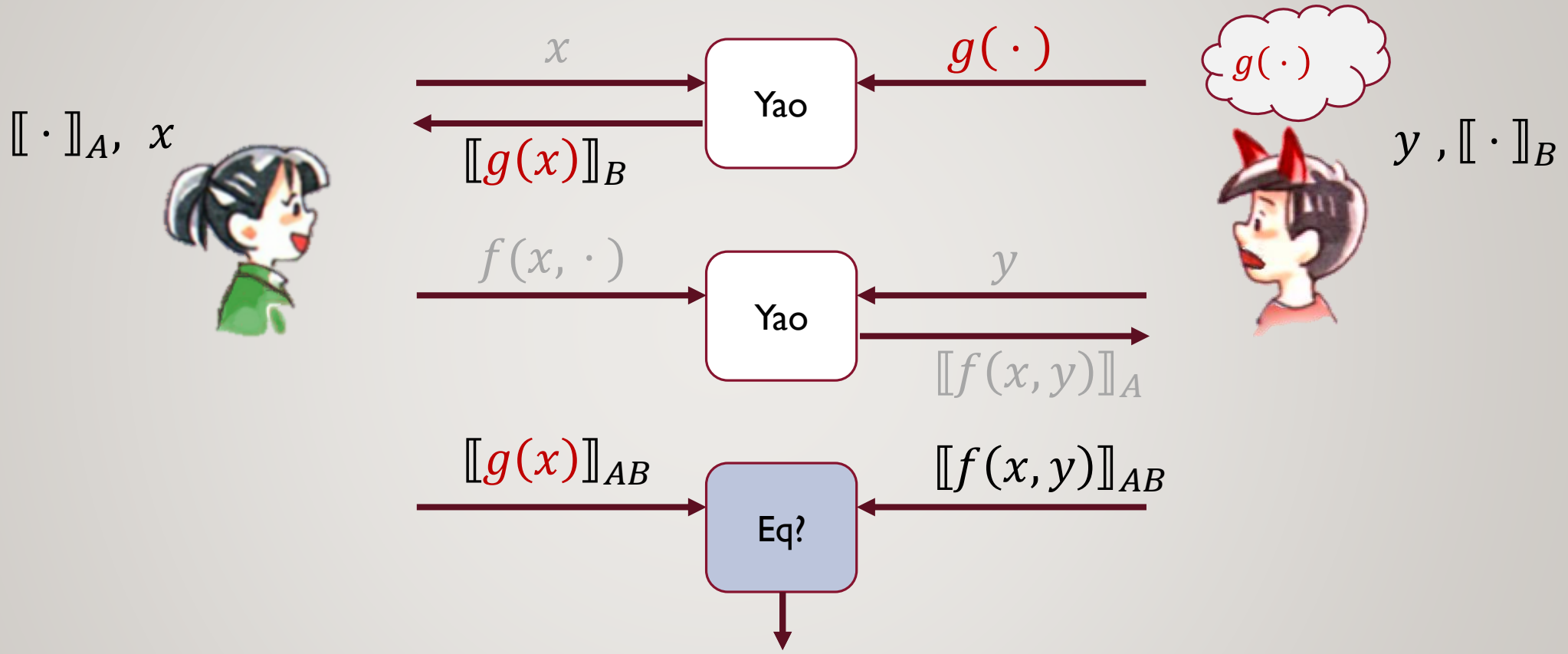
Dual Execution [MohasselFranklin06]



- Define common encoding:

$$[\![z]\!]_{AB} \stackrel{\text{def}}{=} [\![z]\!]_A \oplus [\![z]\!]_B$$

Dual Execution [MohasselFranklin06]



- Define common encoding:

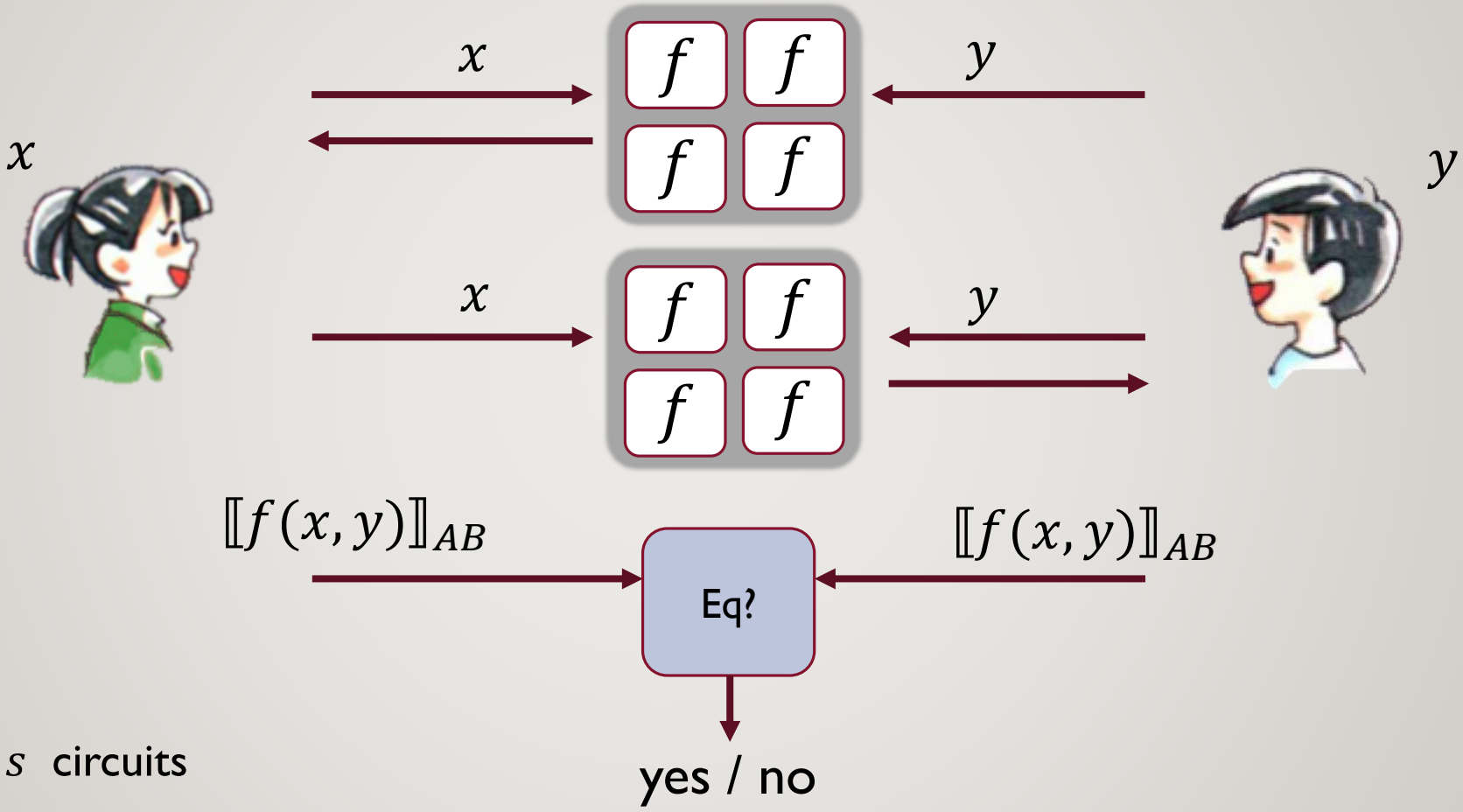
$$[\![z]\!]_{AB} \stackrel{\text{def}}{=} [\![z]\!]_A \oplus [\![z]\!]_B$$

- **Leaks a single bit!**

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

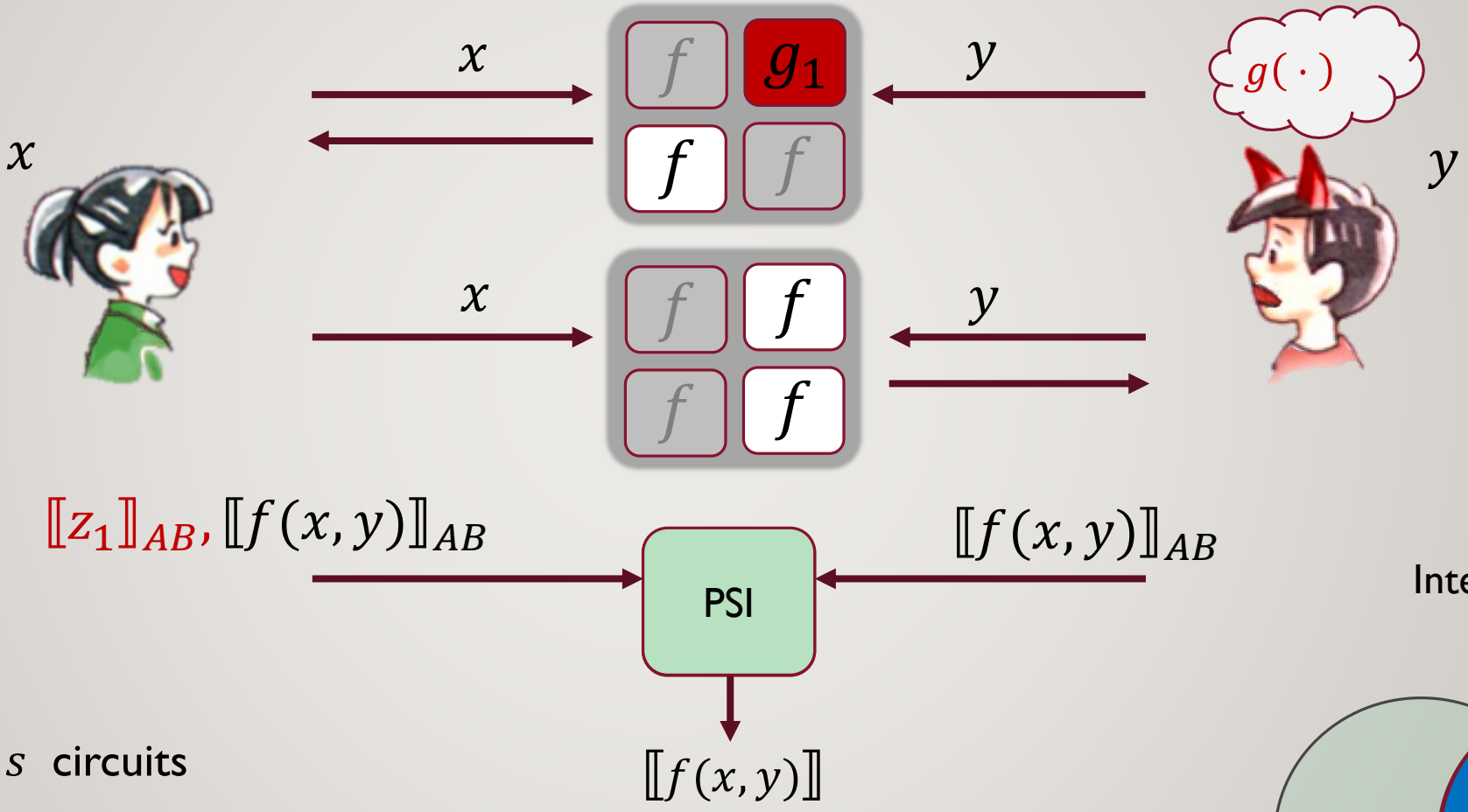
- Guaranteed Correctness

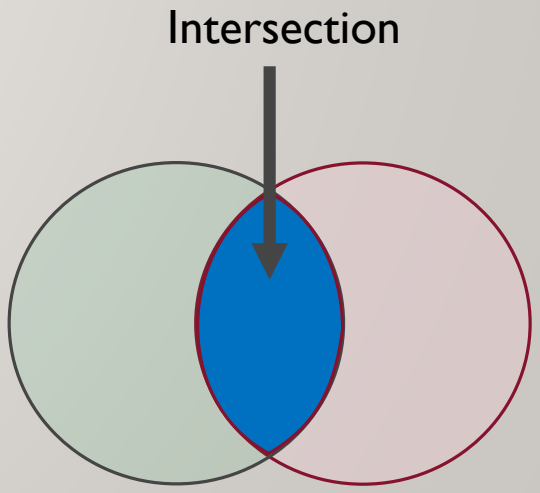
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]



- Send s circuits
- Check some for correctness

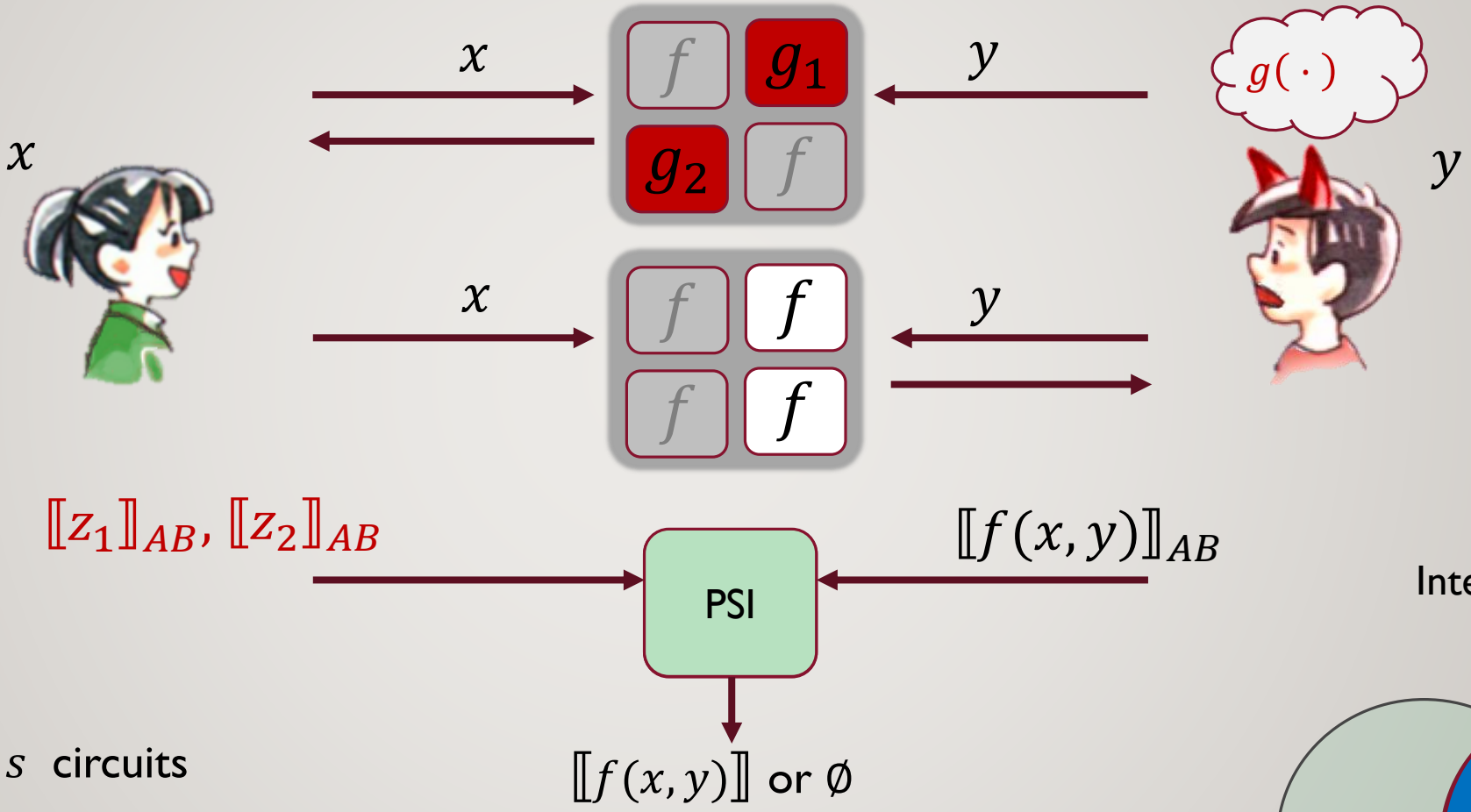
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]



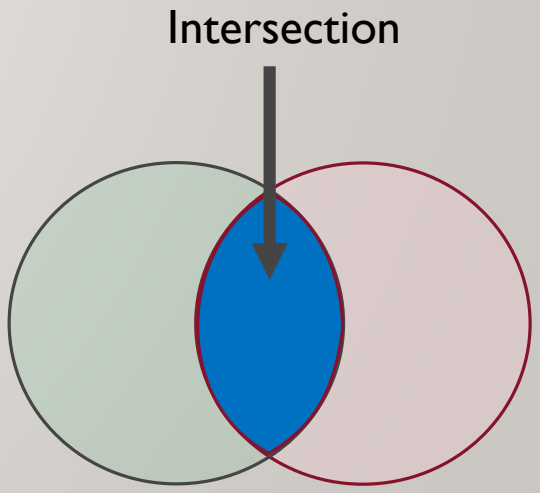


- Send s circuits
- Check some for correctness
- Use Private Set Intersection (PSI) to reconcile

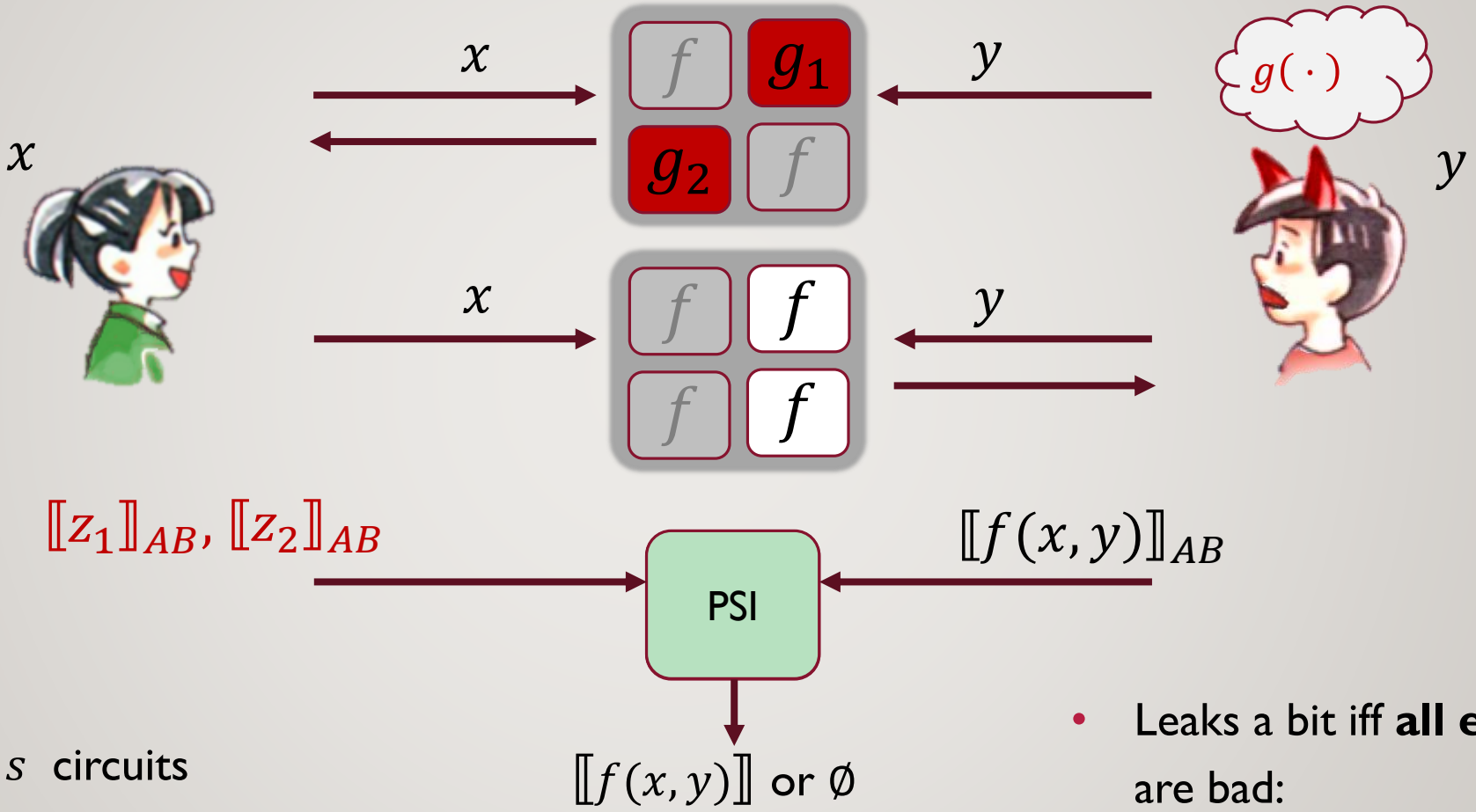
Improved Dual Execution [KolesnikovMohasselRivaRosulek15]



- Send s circuits
- Check some for correctness
- Use Private Set Intersection (PSI) to reconcile



Improved Dual Execution [KolesnikovMohasselRivaRosulek15]



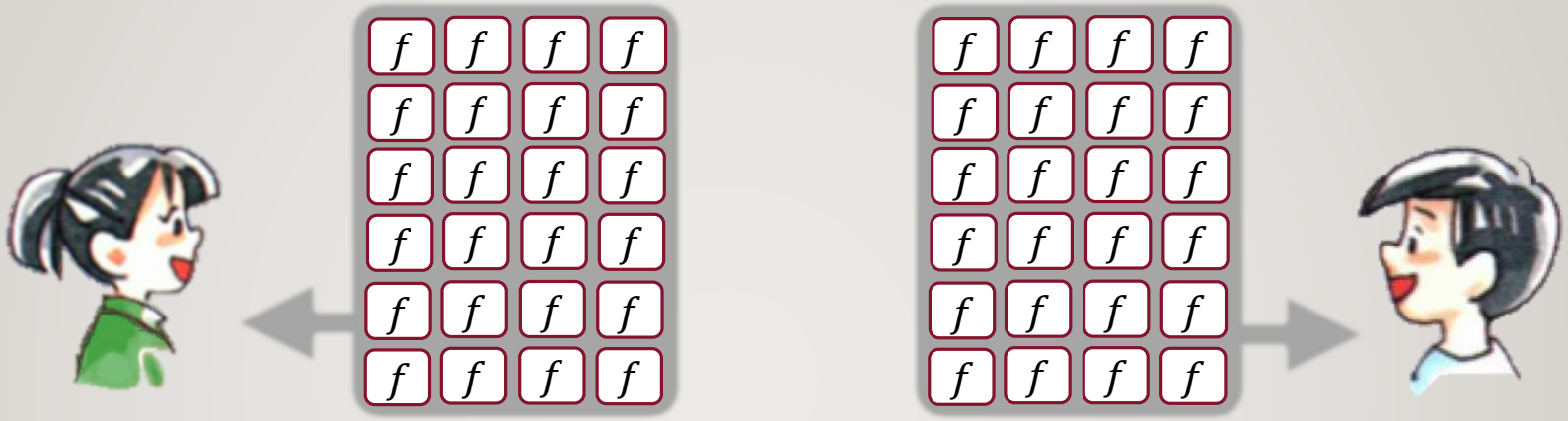
- Send s circuits
- Check some for correctness
- Use Private Set Intersection (PSI) to reconcile

- Leaks a bit iff **all eval. Circuit** are bad:

$$\forall i : g_i(x) \neq f(x,y)$$

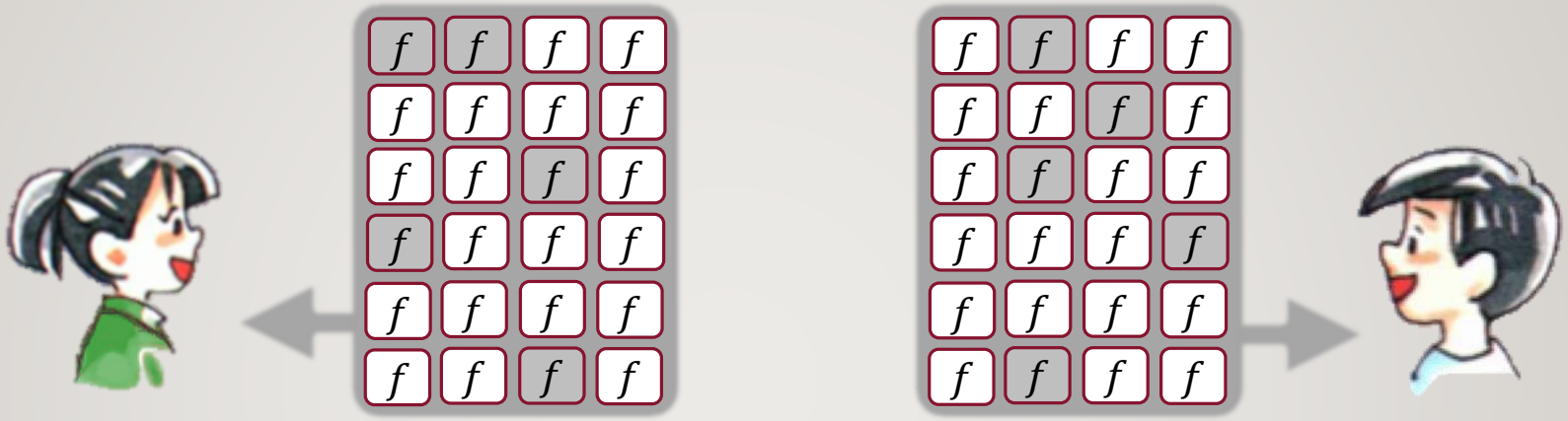
- $\mathbf{Pr}[\boldsymbol{leak\ a\ bit}] = 2^{-s}$

Online – Offline [LindellRiva14,NeilsenOrlandi08,**R**Rosulek16]



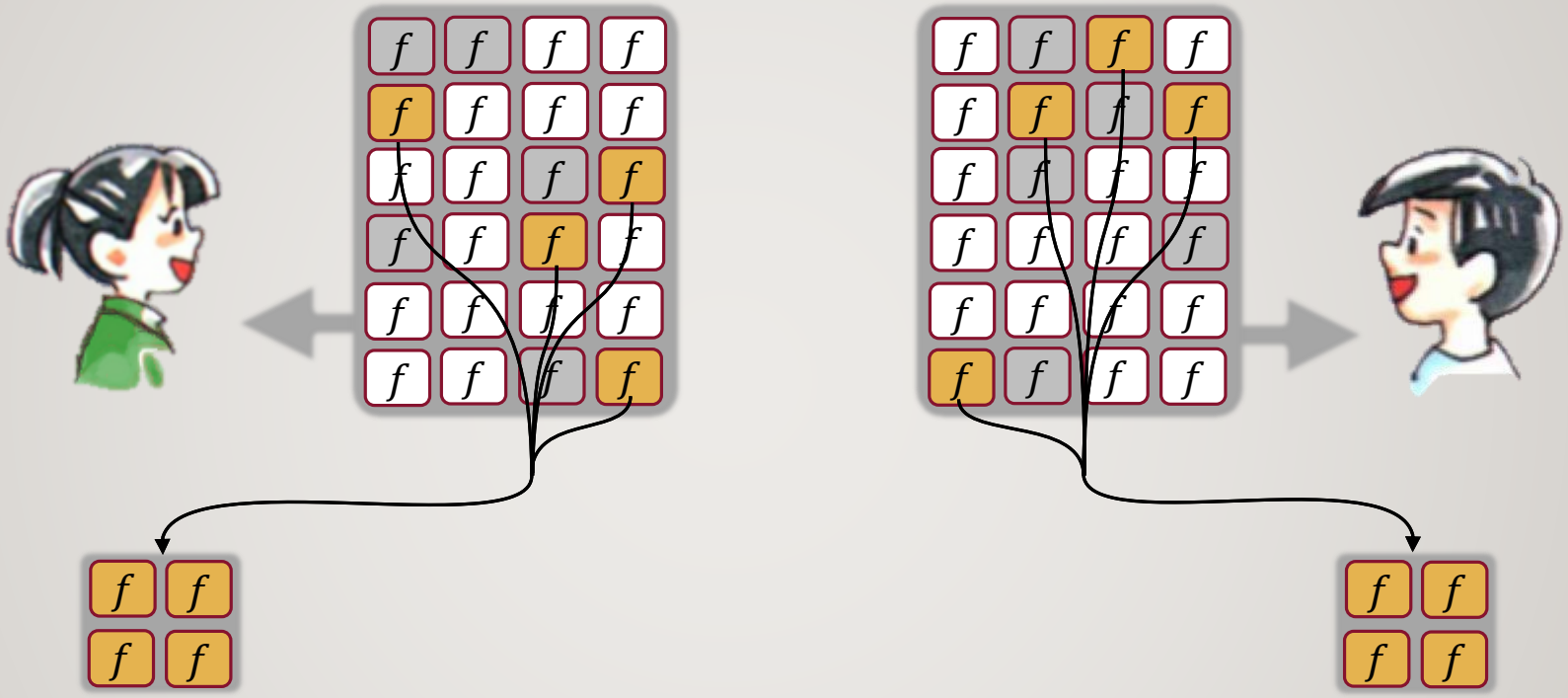
- Want to perform N executions of f
  - Construct enough circuits for all N executions

Online – Offline [LindellRiva14,NeilsenOrlandi08,**R**Rosulek16]



- Want to perform N executions of f
  - Construct enough circuits for all N executions
  - Check some for correctness

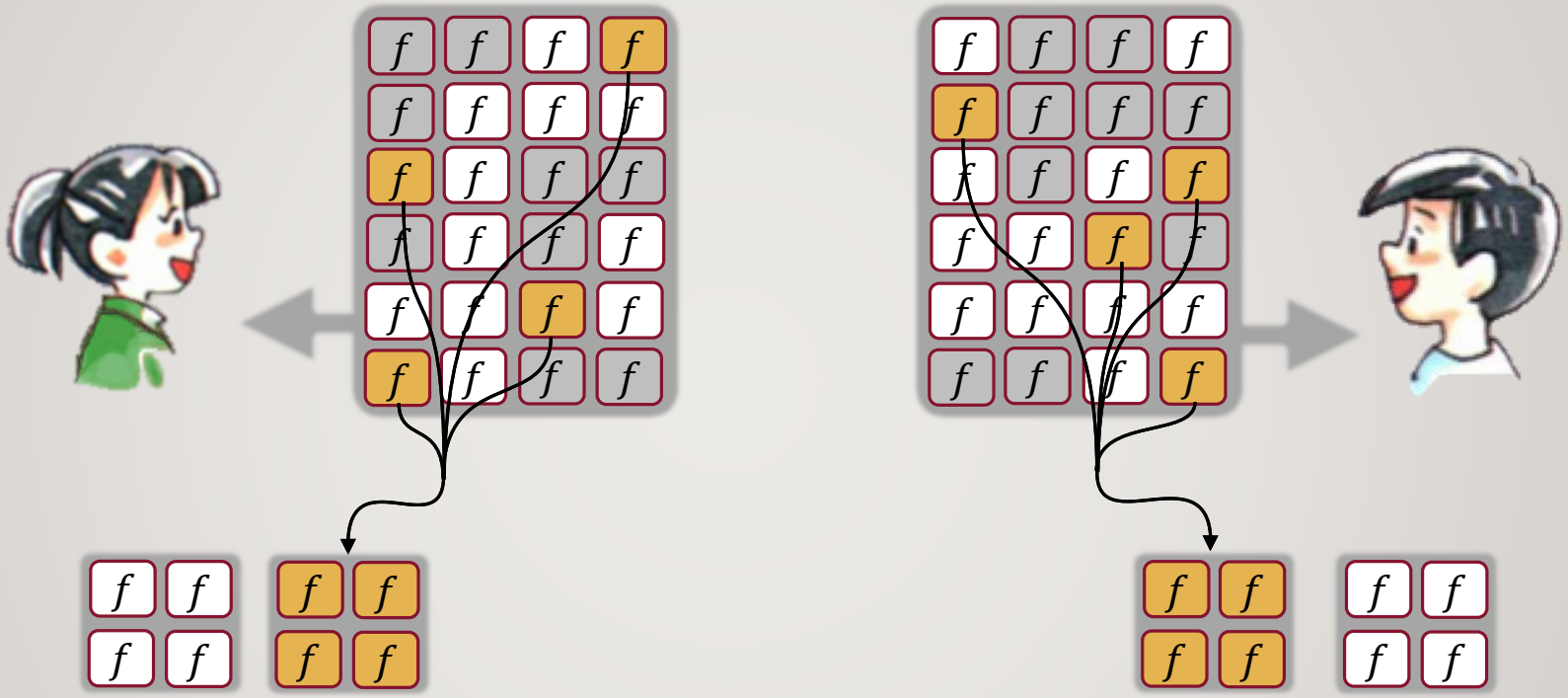
Online – Offline [LindellRiva14,NeilsenOrlandi08,**R**Rosulek16]



- Want to perform N executions of f
  - Construct enough circuits for all N executions
  - Check some for correctness

- Randomly map the rest into bins
  - **Bin size of $s/\log N$ instead of s**
  - E.g. $10\times$ improvement

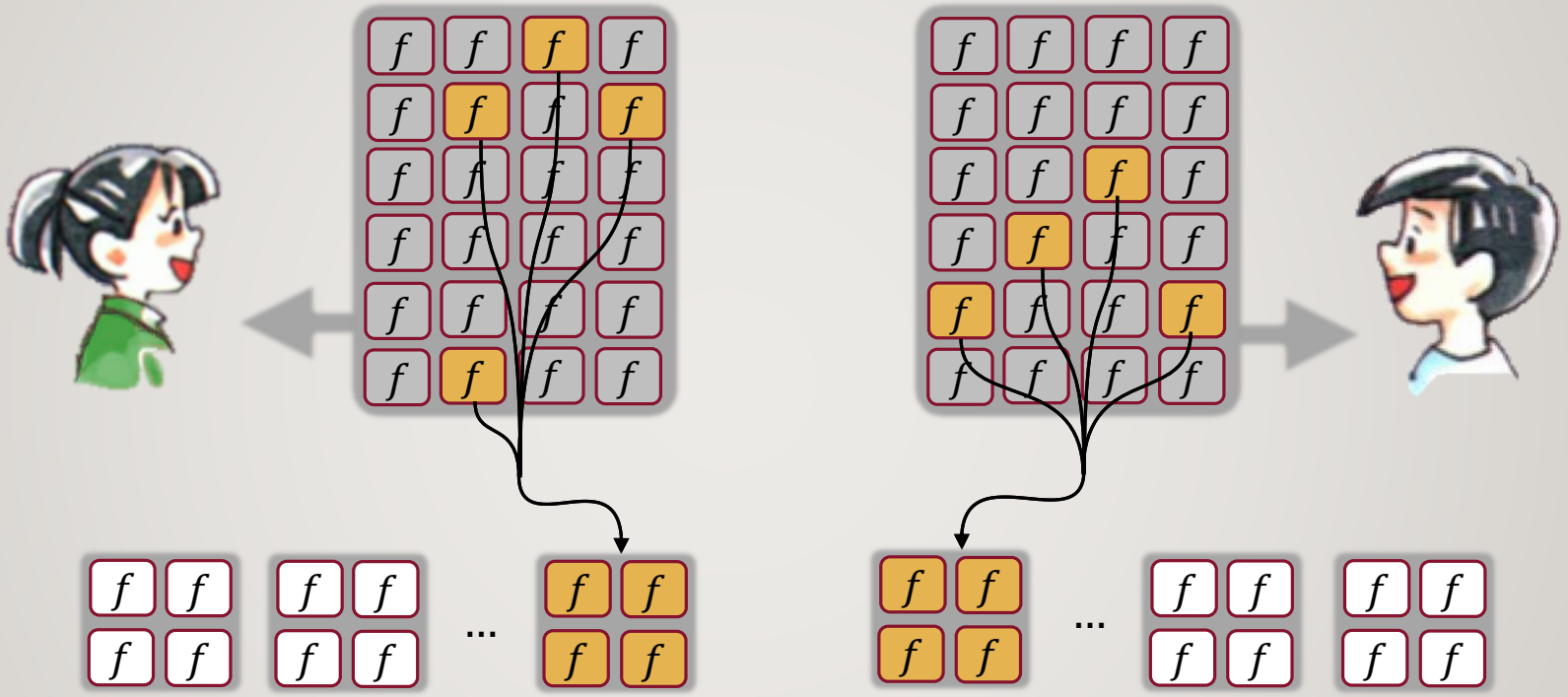
Online – Offline [LindellRiva14,NeilsenOrlandi08,**R**Rosulek16]



- Want to perform N executions of f
  - Construct enough circuits for all N executions
  - Check some for correctness

- Randomly map the rest into bins
  - **Bin size of $s/\log N$ instead of s**
  - E.g. $10\times$ improvement

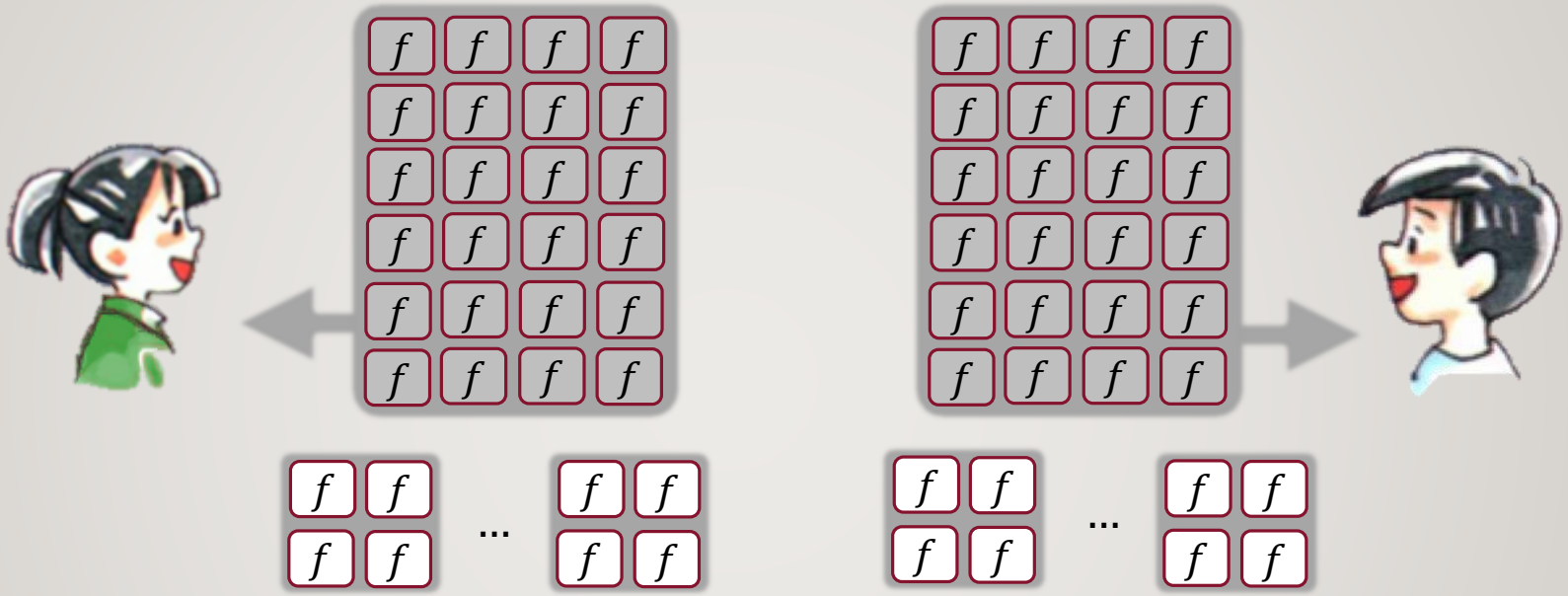
Online – Offline [LindellRiva14,NeilsenOrlandi08,**R**Rosulek16]



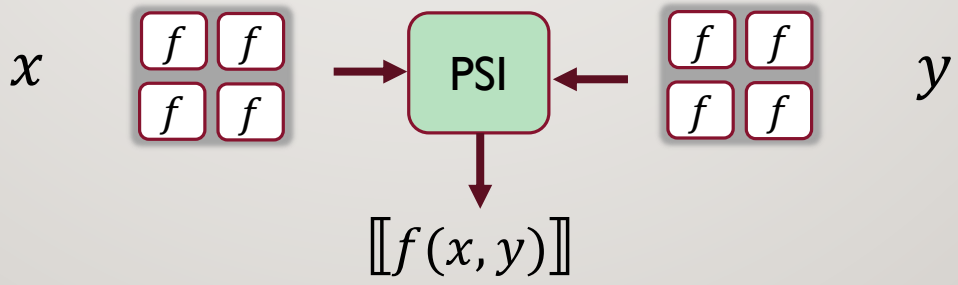
- Want to perform N executions of f
  - Construct enough circuits for all N executions
  - Check some for correctness

- Randomly map the rest into bins
  - **Bin size of $s/\log N$ instead of s**
  - E.g. $10\times$ improvement

Online – Offline [LindellRiva14,NeilsenOrlandi08,**R**Rosulek16]

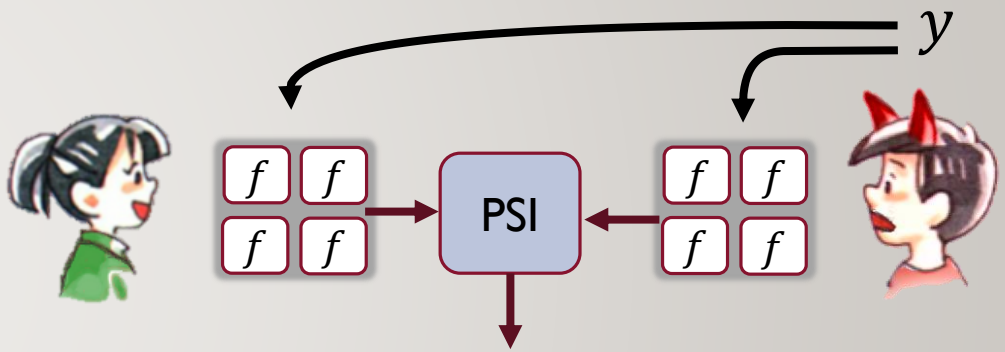


- Use one bin per evaluation



Challenge #1: Input Consistency [RRosulek16]

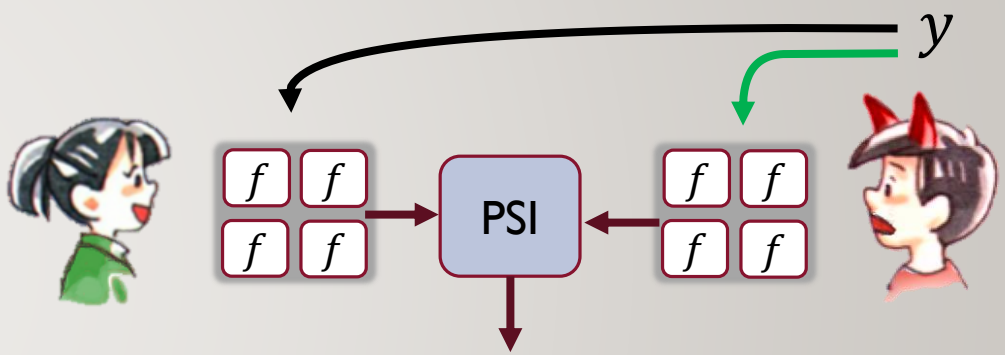
How to ensure Bob used the same y in all circuits?



Challenge #1: Input Consistency [RRosulek16]

How to ensure Bob used the same y in all circuits?

- Circuit generated by Alice
  - Bob receives input via OT [easy]

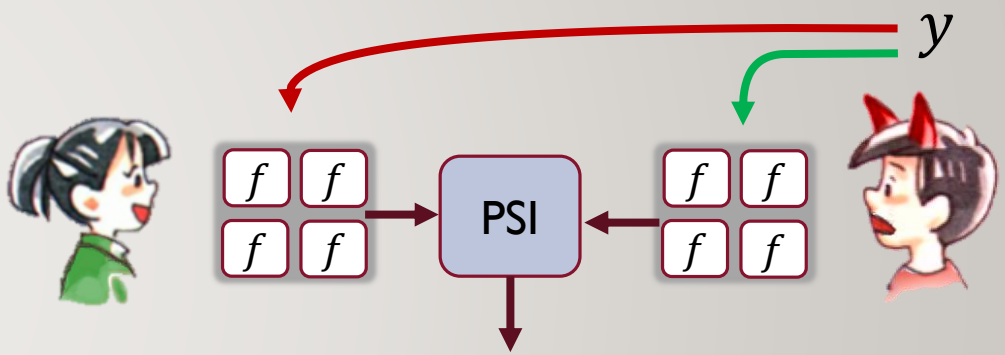


Challenge #1: Input Consistency [RRosulek16]

How to ensure Bob used the same y in all circuits?

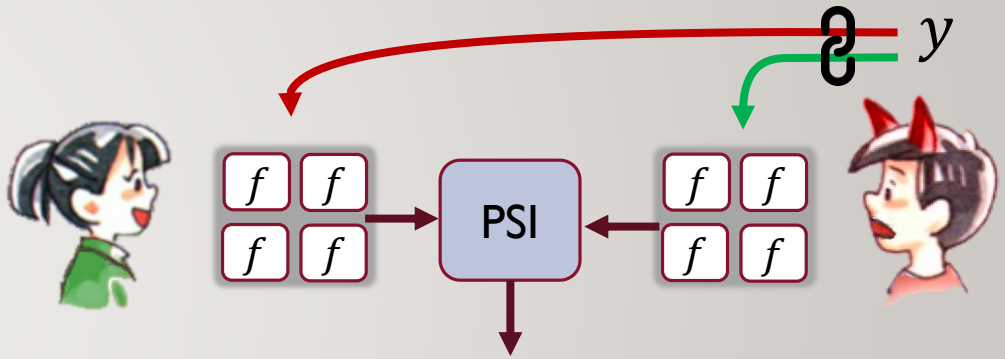
- Circuit generated by Alice
  - Bob receives input via OT [easy]

- Circuit generated by Bob [hard]



Challenge #1: Input Consistency [RRosulek16]

How to ensure Bob used the same y in all circuits?

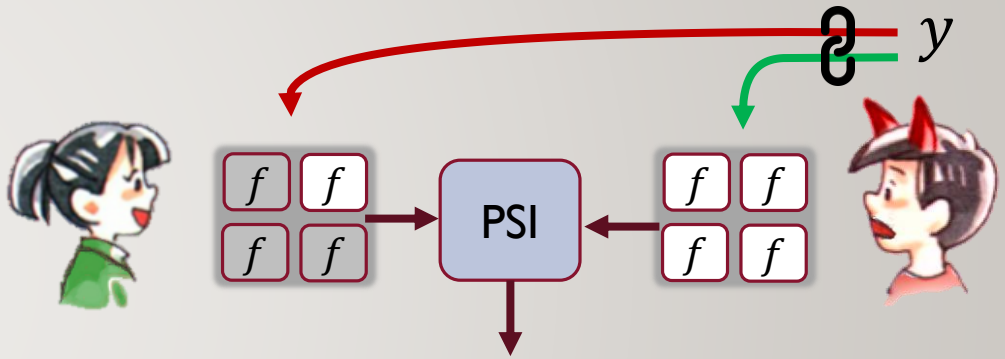


- Circuit generated by Alice
  - Bob receives input via OT [easy]

- Circuit generated by Bob [hard]
  - In the offline, Bob tells Alice the relationship ∾ between the two arrows
  - Alice check ∾ in the cut and choose

Challenge #1: Input Consistency [RRosulek16]

How to ensure Bob used the same y in all circuits?



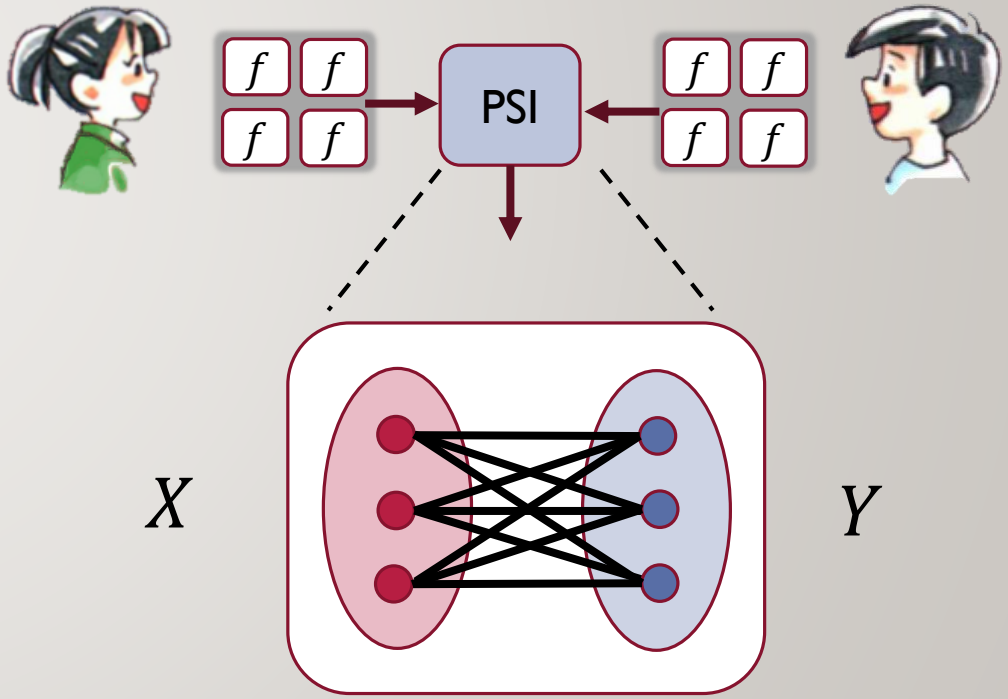
- Circuit generated by Alice
  - Bob receives input via OT [easy]

- Circuit generated by Bob [hard]
  - In the offline, Bob tells Alice the relationship ∾ between the two arrows
  - Alice check ∾ in the cut and choose

  - Consistent with the relationship ⇒ at least of one of Bob's circuits uses y
    - Requires **no crypto operations**

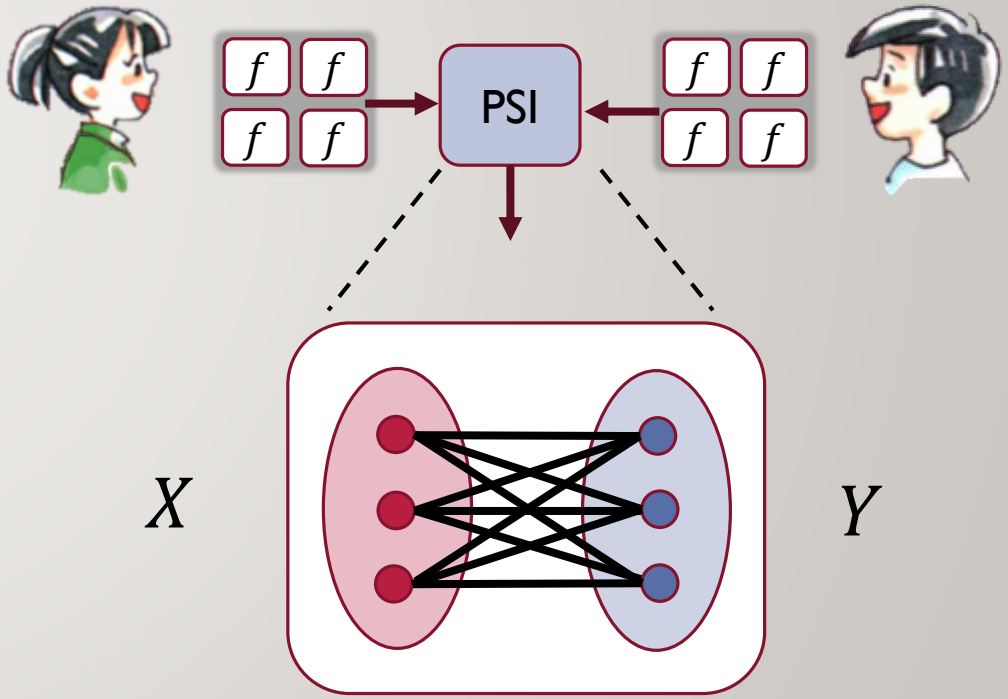
Challenge #2: Private Set Intersection (PSI) [RRosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol



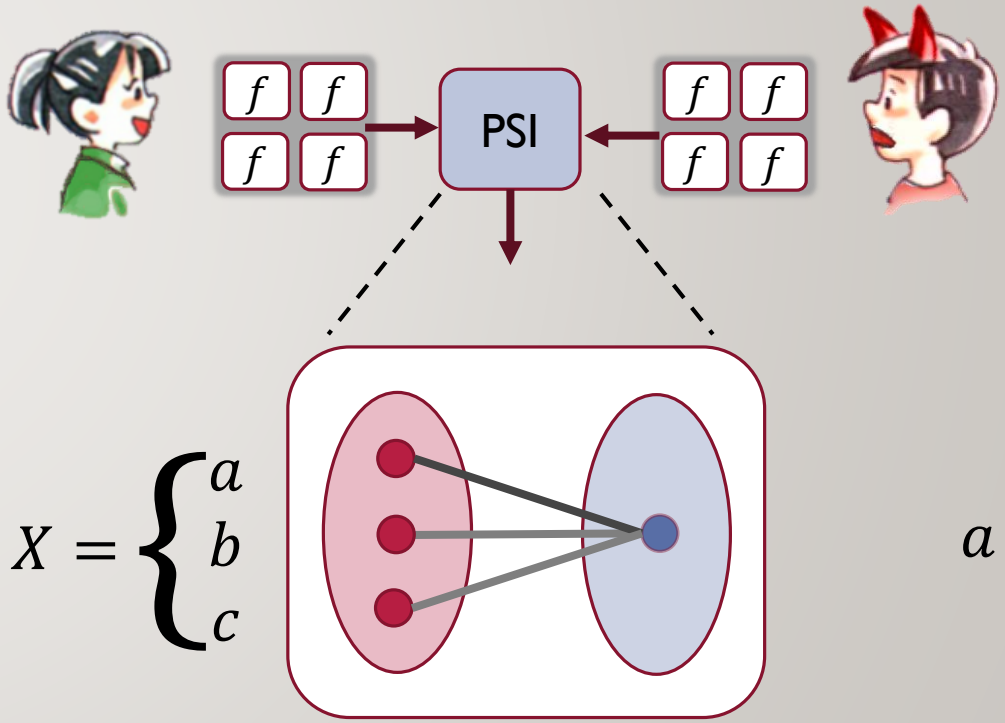
Challenge #2: Private Set Intersection (PSI) [RRosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
- Issues: Not malicious secure in general
  - Can not be simulated



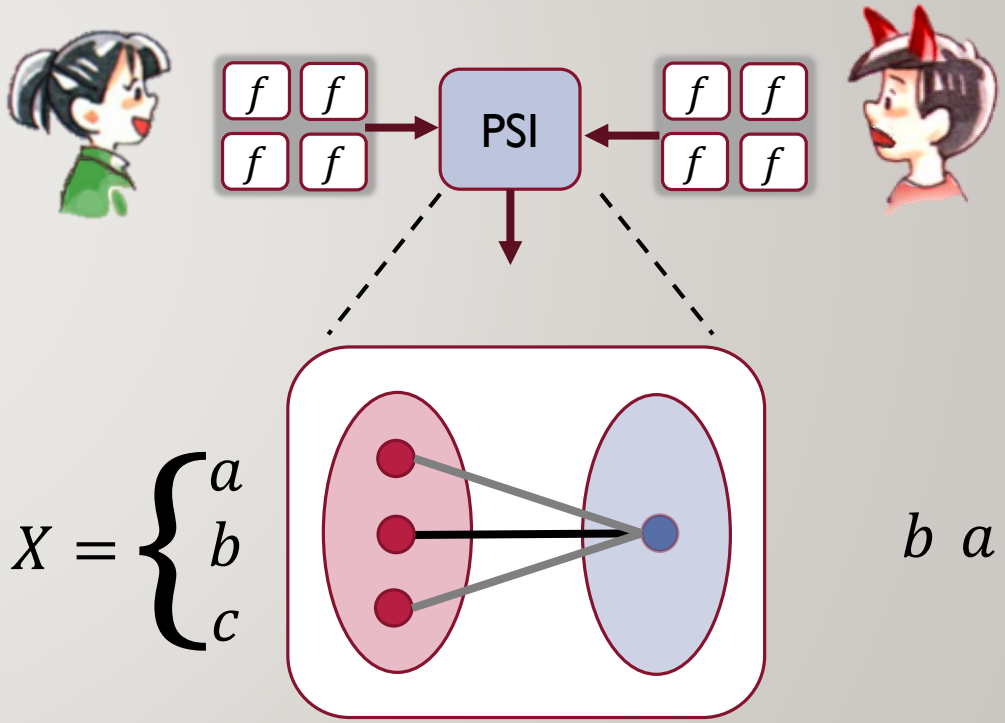
Challenge #2: Private Set Intersection (PSI) [RRosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
- Issues: Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set



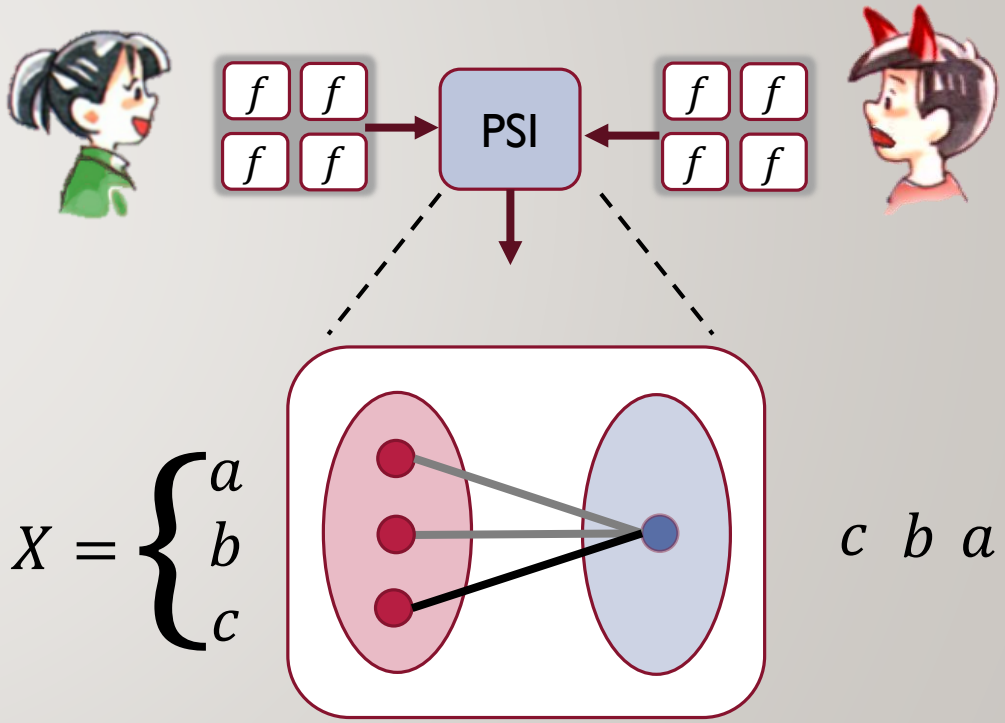
Challenge #2: Private Set Intersection (PSI) [RRosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
- Issues: Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set



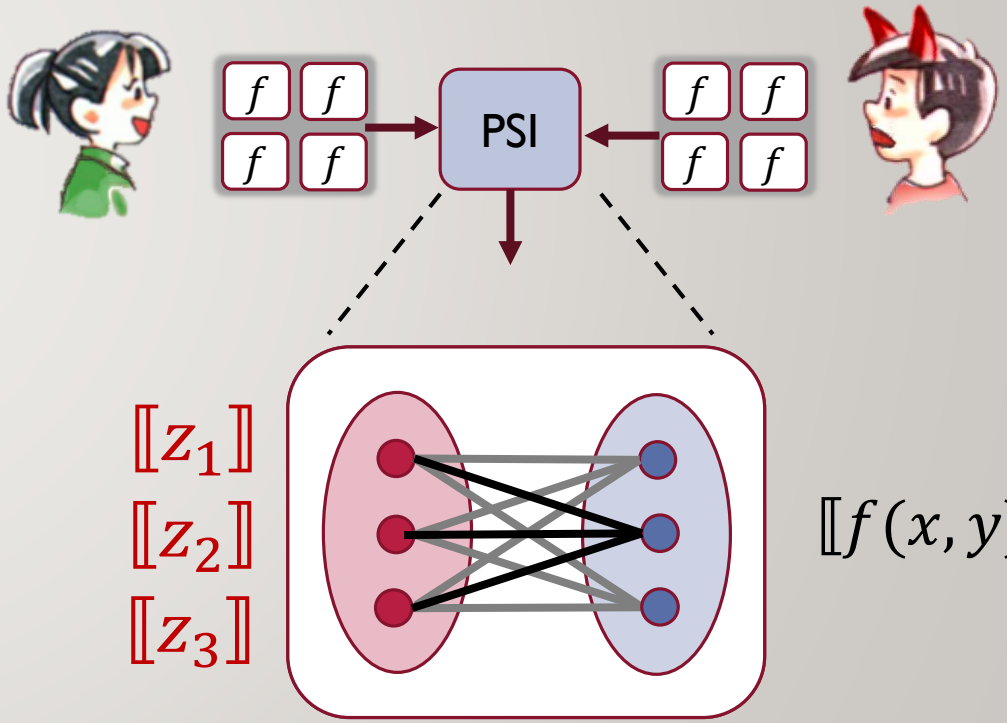
Challenge #2: Private Set Intersection (PSI) [RRosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
- Issues: Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set



Challenge #2: Private Set Intersection (PSI) [RRosulek16]

- Build PSI from Private Equality Test [PinkasSchneiderZohner14]
  - Fastest PSI protocol
- Issues: Not malicious secure in general
  - Can not be simulated
  - Ex: singleton set



- Ideal: Bob only knows one valid PSI input

$$[\![f(x,y)]\!]$$

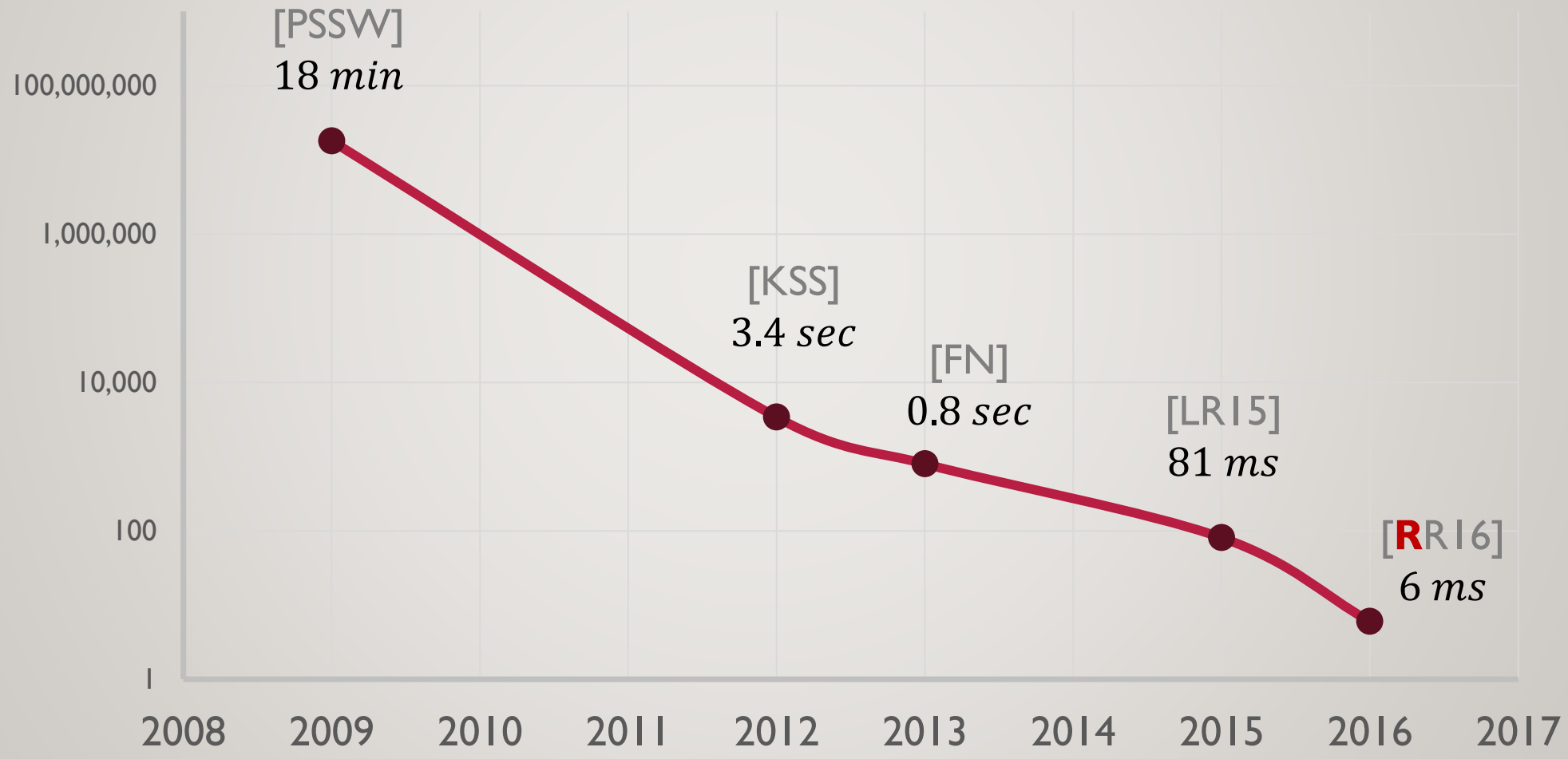
- Simulator **doesn't need to extract Bob input!**
  - Just test if it contains $[\![f(x,y)]\!]$

Performance

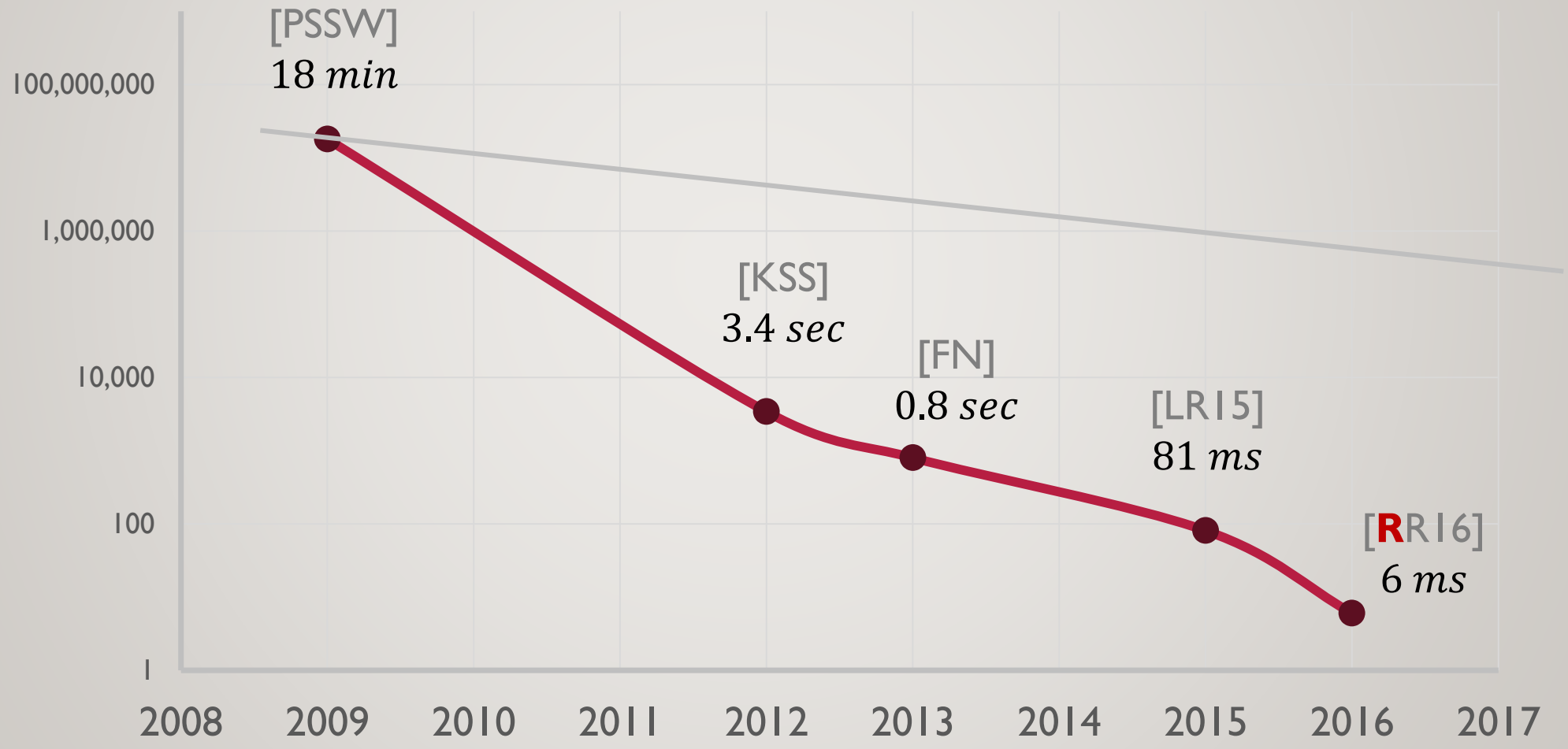
Function	[RRosulek16]		[LindellRiva15]		[DamgårdZakarias15]	
	Offline	Online	Offline	Online	Offline	Online
AES	$\mathbf{5.1\ ms}$	$\mathbf{1.3\ ms}$	$74\ ms$	$7\ ms$	high?	$6\ ms$
SHA-256	$\mathbf{48.0\ ms}$	$\mathbf{8.1\ ms}$	$206\ ms$	$33\ ms$	-	-

- Amortized cost for $N = 1{,}024$ evaluations
  - Amazon `c4.8xLarge` = 36 core, 64GB RAM
  - Statistical security $\kappa = 40$
- Maximum **throughput**: $0.26\ ms$ / AES block (3800+ Hz)
  - [DamgårdZakarias15] report $0.4\ ms$

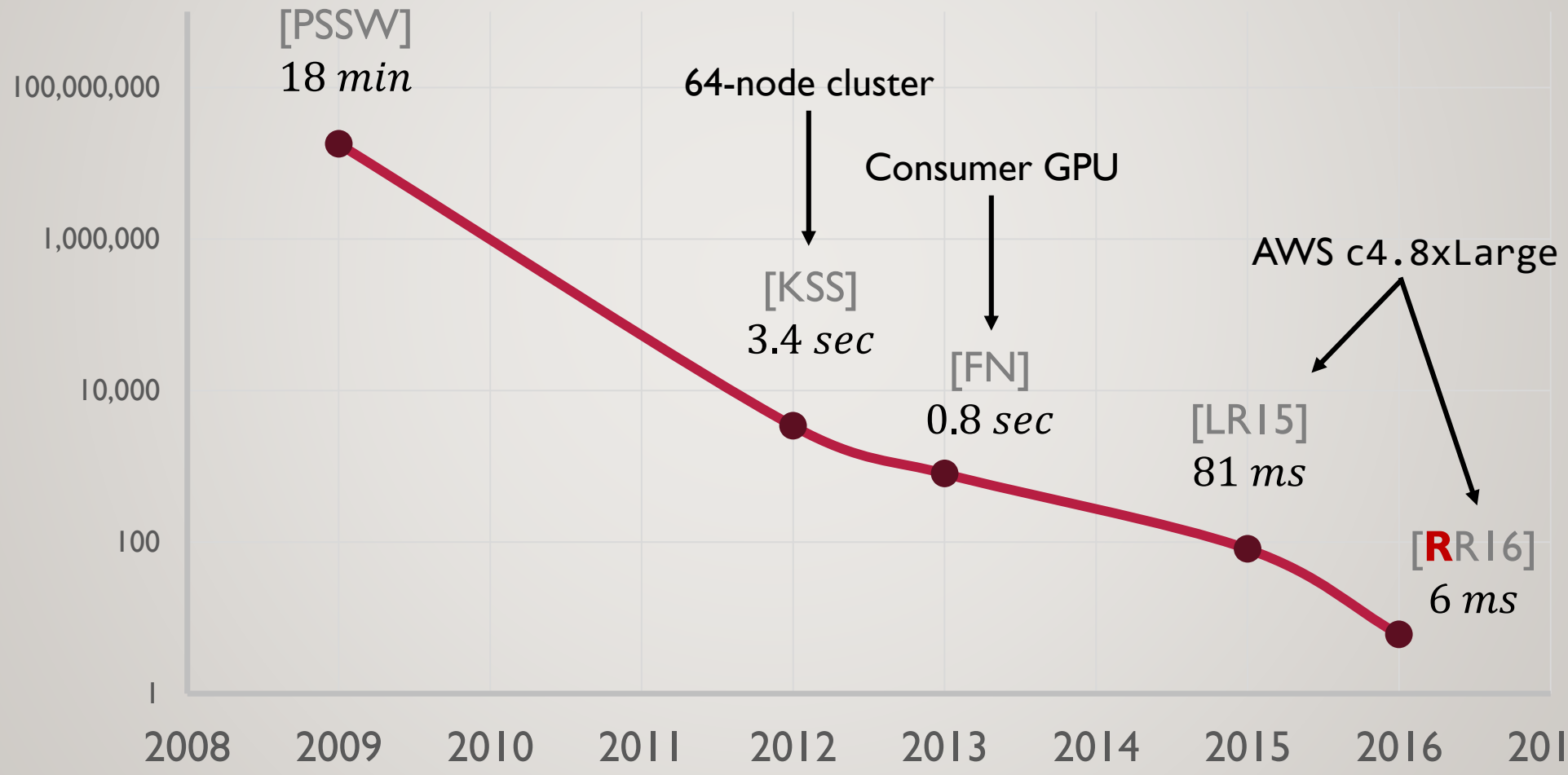
Total Protocol Times for AES



Total Protocol Times for AES



Total Protocol Times for AES



Summary

- Online-offline dual execution
  - Faster 2PC with malicious security to date: $1.3ms$ AES
  - Some security advantages over "classic" cut-and-choose

- Future Work:
  - Hybrid protocols: combine [**R**Rosulek16] with [DamgårdZakarias15]
    - fast offline
    - function independent offline
  - Transfer advances from online-offline to single execution setting

The End

Thanks

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

github.com/osu-crypto/batchDualEx

Peter Rindal
Mike Rosulek

Oregon State UNIVERSITY OSU