List Oblivious Transfer and Applications to Round-Optimal Black-Box Multiparty Coin Tossing

Michele Ciampi

The University of Edinburgh

Rafail Ostrovsky

UCLA

Luisa Siniscalchi

Technical University of Denmark

Hendrik Waldner

UMD & MPI-SP

Multiparty Computation (MPC)

$f(x_1,x_2,x_3,x_4)$

x_1

x_2

$f(x_1,x_2,x_3,x_4)$

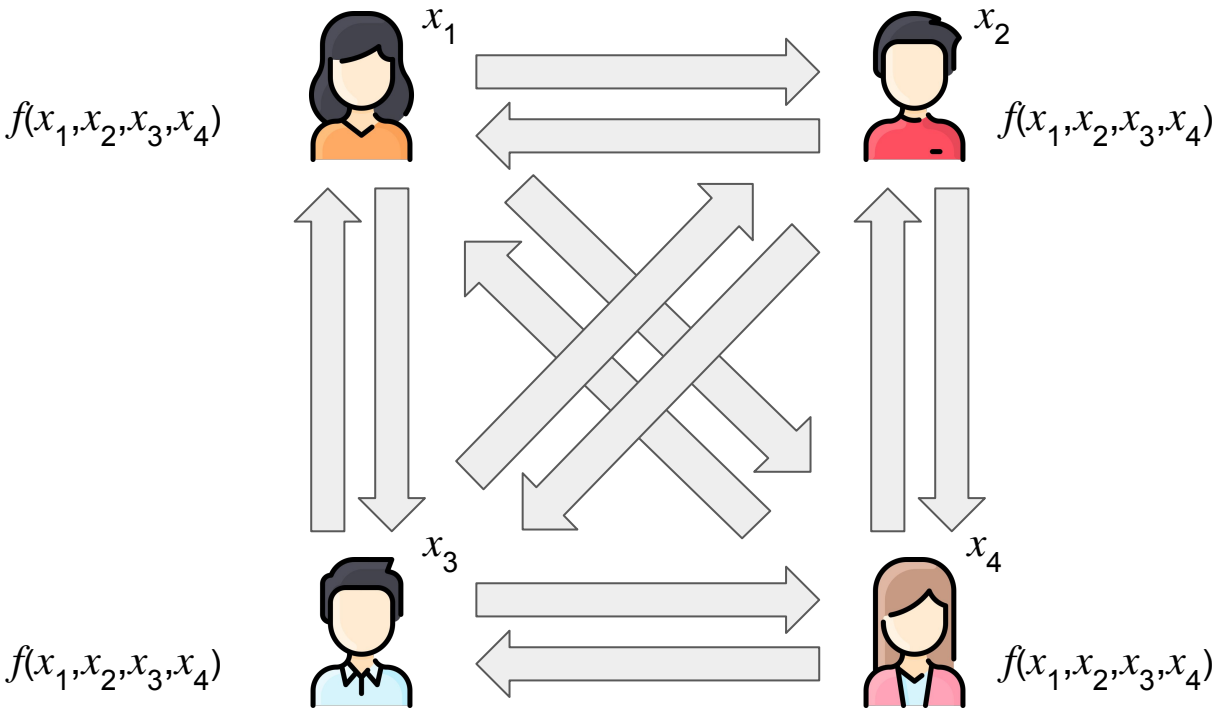
$f(x_1,x_2,x_3,x_4)$

x_3

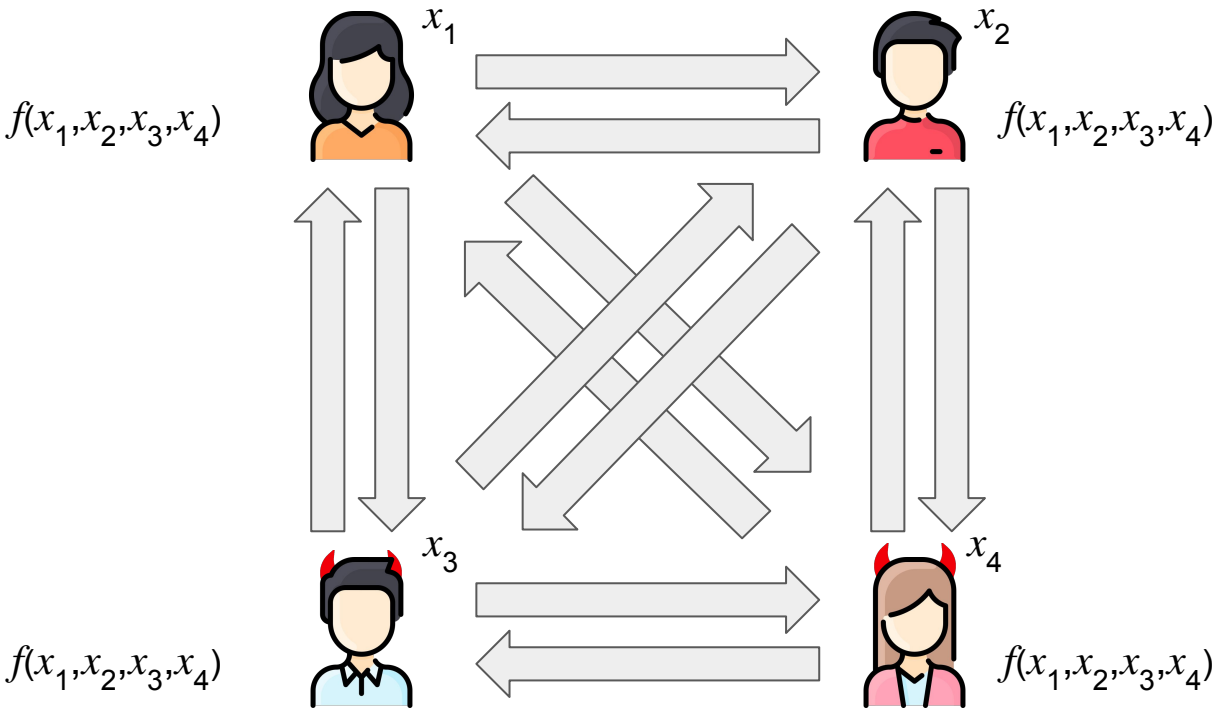
x_4

$f(x_1,x_2,x_3,x_4)$

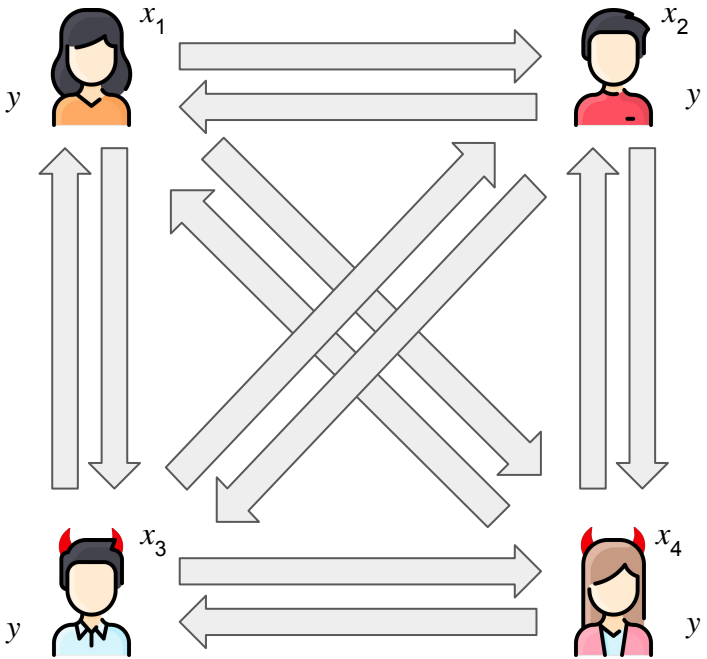
Multiparty Computation (MPC)



Multiparty Computation (MPC)

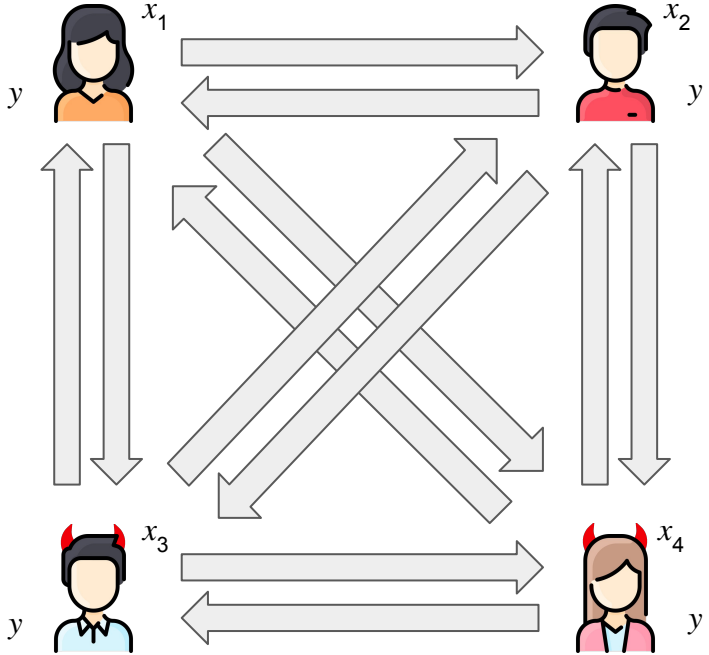


Multiparty Computation (MPC)

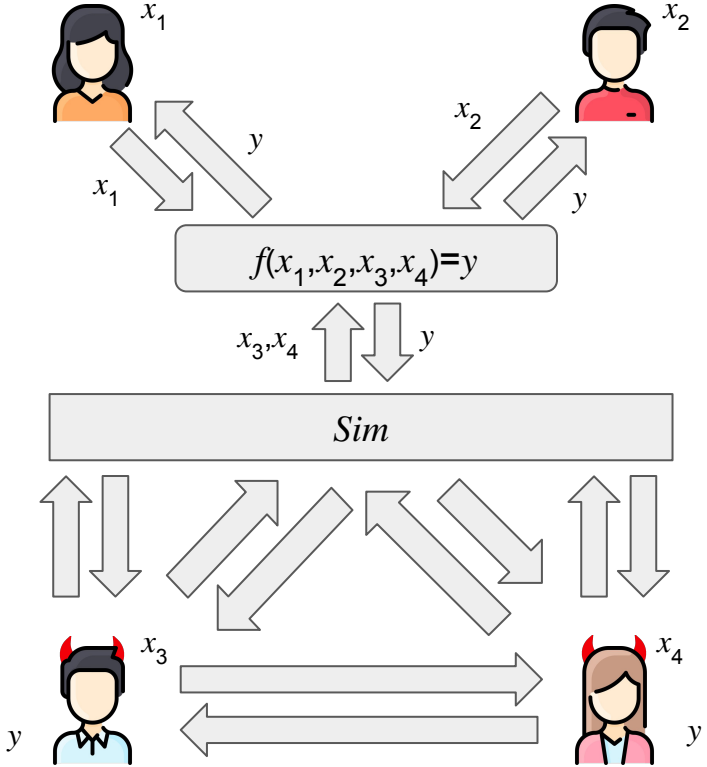


$\approx$

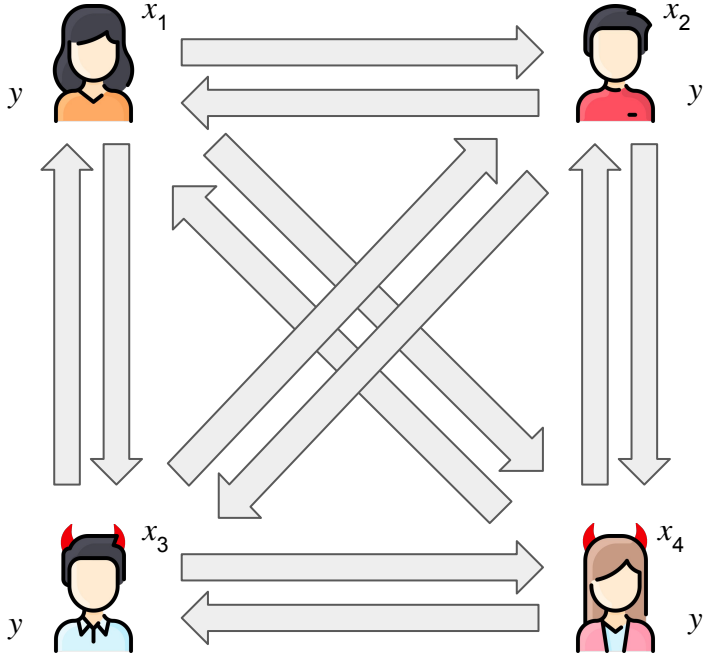
Multiparty Computation (MPC)



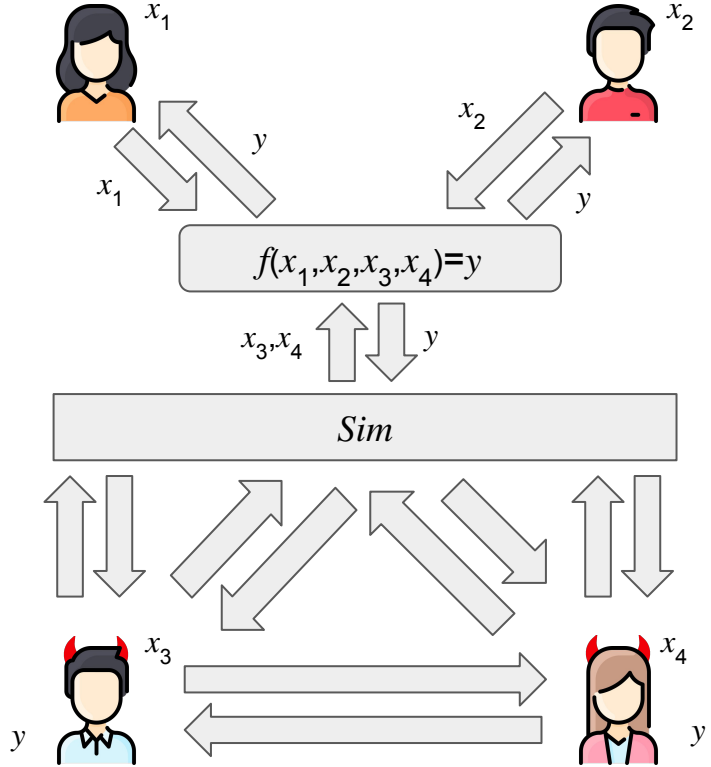
≈



Multiparty Computation (MPC)



≈



Malicious Security in the Plain Model with Black-Box simulation

Round-Optimal Black-Box MPC

Round-Optimal Black-Box MPC

Round-Optimal:

⇒ 4 rounds are necessary [GMPP16] and sufficient [BGJ+18,HHPV18,CCG+20]

Round-Optimal Black-Box MPC

Round-Optimal:

⇒ 4 rounds are necessary [GMPP16] and sufficient [BGJ+18,HHPV18,CCG+20]

Black-Box:

⇒ Primitives are only used in a black-box way

Round-Optimal Black-Box MPC

Round-Optimal:

⇒ 4 rounds are necessary [GMPP16] and sufficient [BGJ+18,HHPV18,CCG+20]

Black-Box:

⇒ Primitives are only used in a black-box way

Current State: 5 rounds [IKSS21] and 4 rounds from subexp secure assumptions [IKSS23]

Round-Optimal Black-Box MPC

Round-Optimal:

⇒ 4 rounds are necessary [GMPP16] and sufficient [BGJ+18,HHPV18,CCG+20]

Black-Box:

⇒ Primitives are only used in a black-box way

Current State: 5 rounds [IKSS21] and 4 rounds from subexp secure assumptions [IKSS23]

In this work: 4 round black-box secure MPC for input-less functionalities
& general functionalities against non-aborting adversaries

Round-Optimal Black-Box MPC

Round-Optimal:

⇒ 4 rounds are necessary [GMPP16] and sufficient [BGJ+18,HHPV18,CCG+20]

Black-Box:

⇒ Primitives are only used in a black-box way

Current State: 5 rounds [IKSS21] and 4 rounds from subexp secure assumptions [IKSS23]

In this work: 4 round black-box secure MPC for input-less functionalities
& general functionalities against non-aborting adversaries

Primitives used in [IPS08,IKSS21]

Primitives used in [IPS08,IKSS21]

- semi-malicious 4 round MPC protocol

Primitives used in [IPS08,IKSS21]

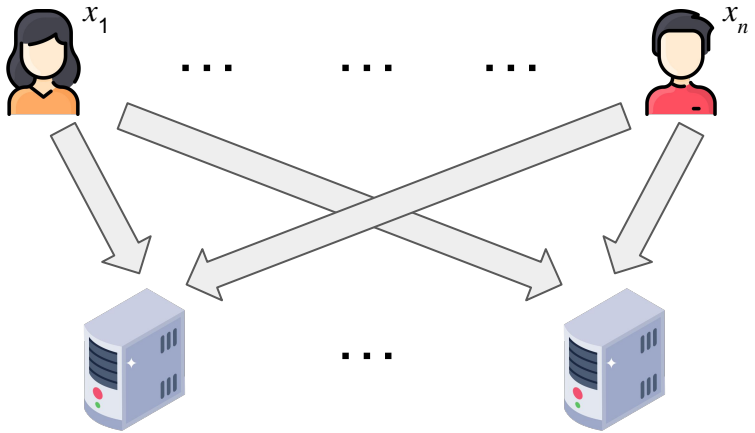
- semi-malicious 4 round MPC protocol
- 2 round n-client m-server protocol

Primitives used in [IPS08,IKSS21]

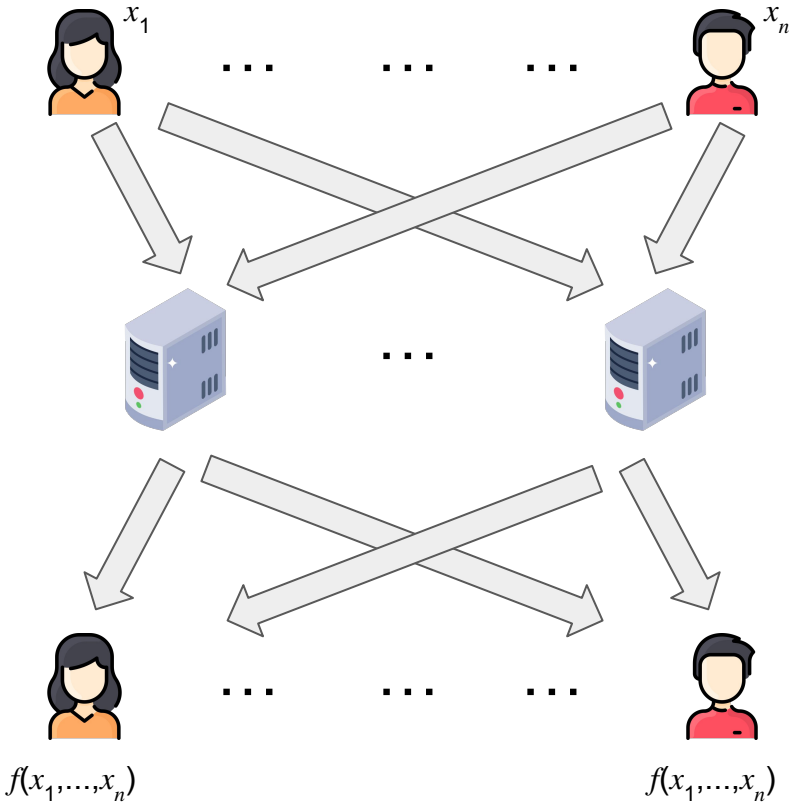
- semi-malicious 4 round MPC protocol
- 2 round n-client m-server protocol
- 4 round non-malleable k-out-of-n OT

n-Client m-Server Protocol

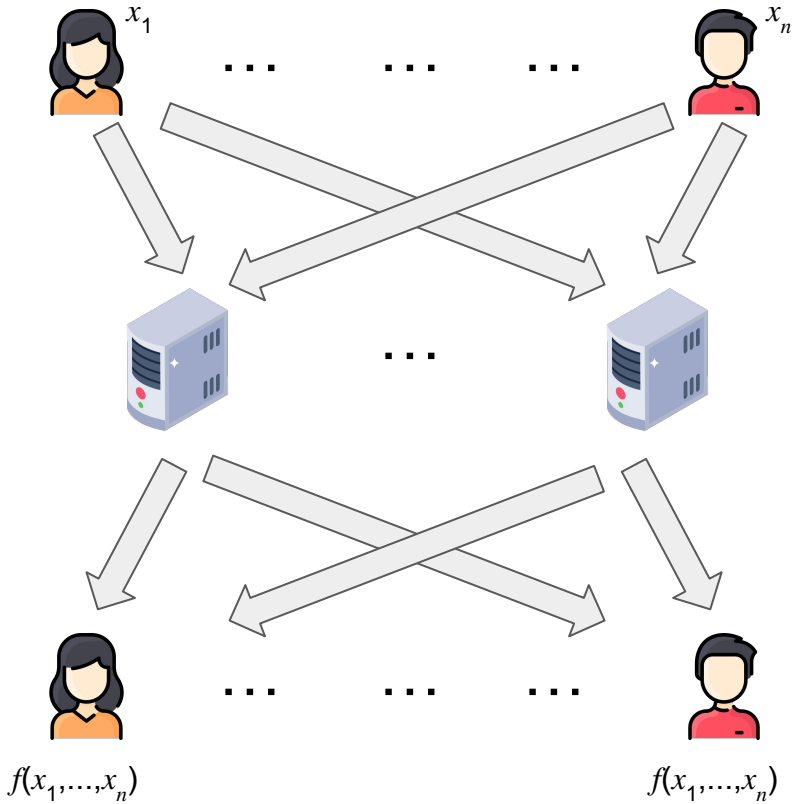
n-Client m-Server Protocol



n-Client m-Server Protocol



n-Client m-Server Protocol

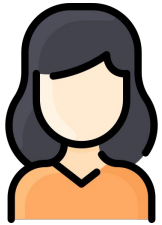


secure against adaptive, malicious adversaries corrupting (m-1)/3 servers

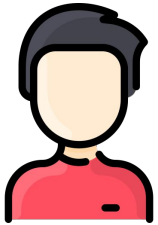
k-out-of-n OT

Receiver

S, $|S|=k$

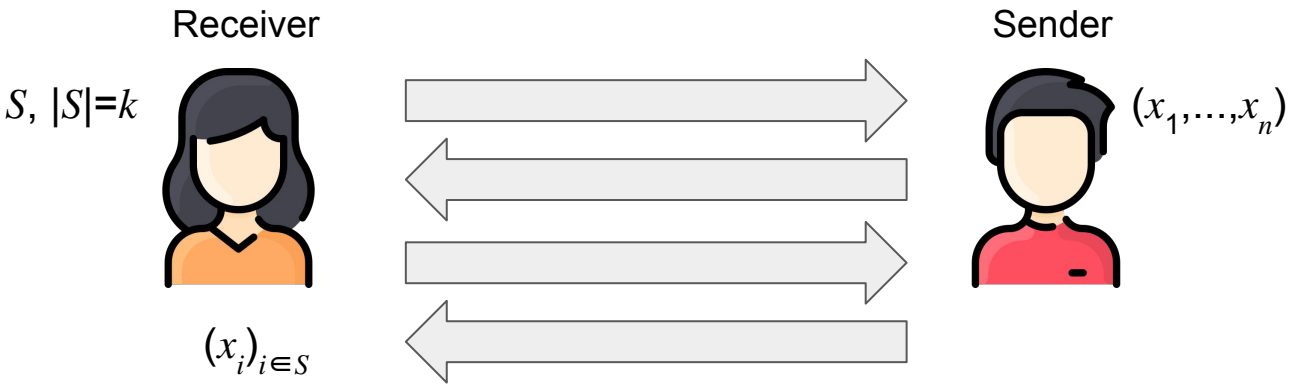


Sender

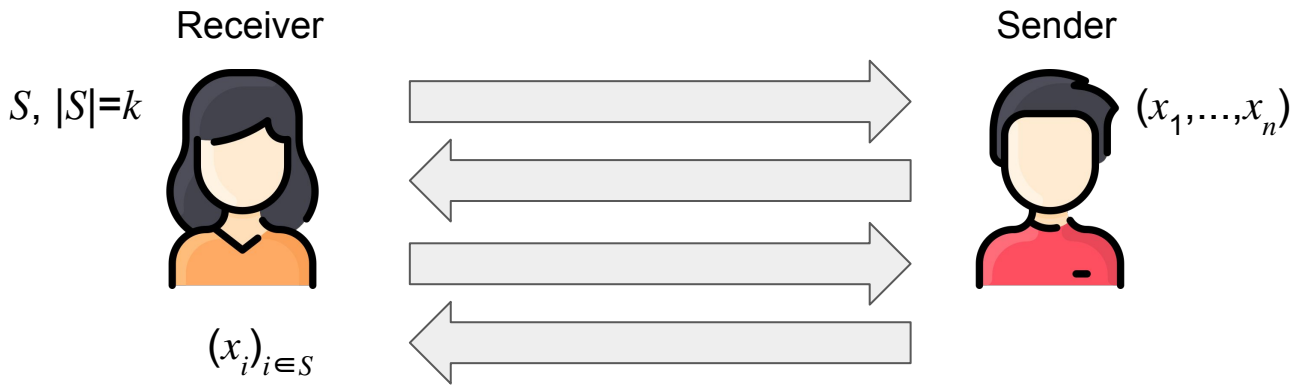


$(x_1,\ldots,x_n)$

k-out-of-n OT

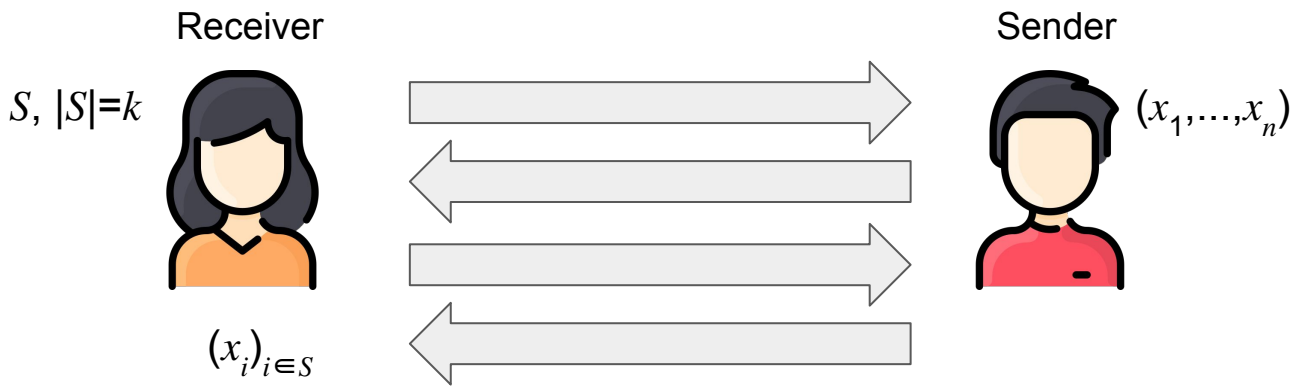


k-out-of-n OT



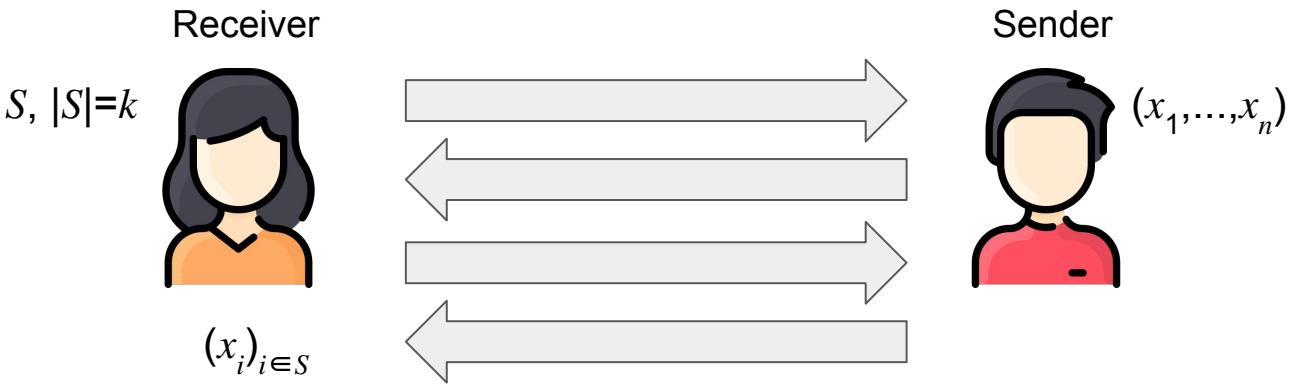
pairwise execution of k-out-of-n OT between all the participating parties→Watchlist

k-out-of-n OT



pairwise execution of k-out-of-n OT between all the participating parties→Watchlist
⇒ requires non-malleability

k-out-of-n OT



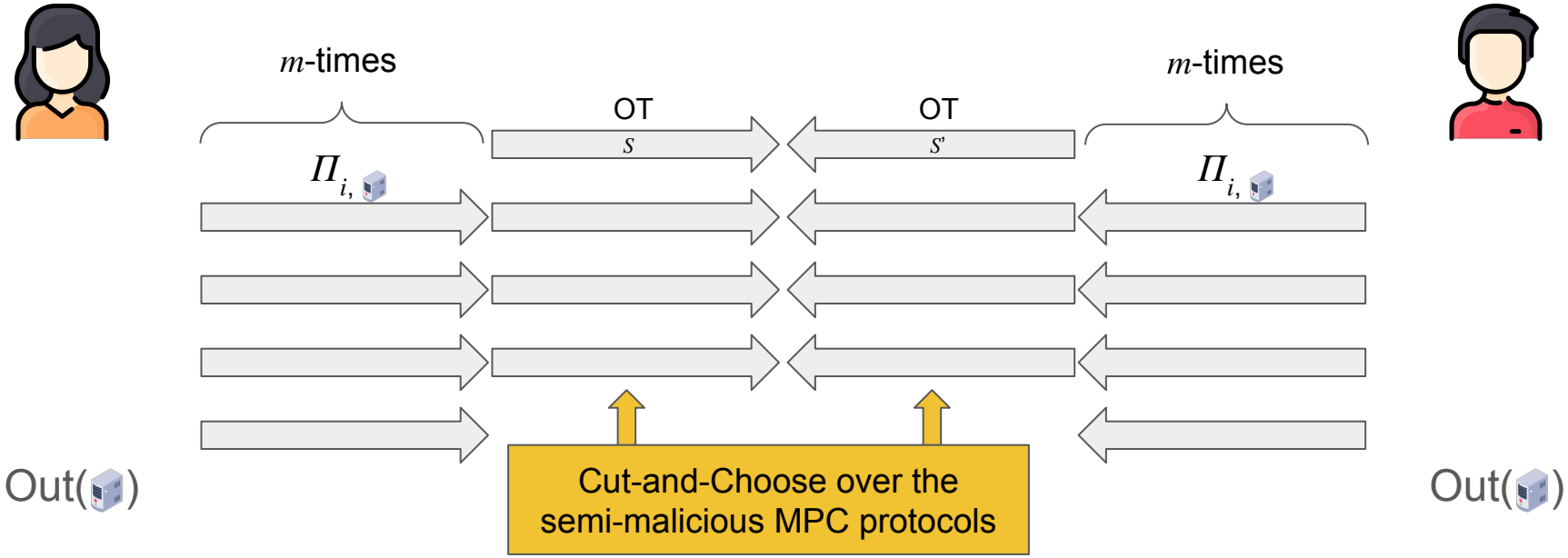
pairwise execution of k-out-of-n OT between all the participating parties→Watchlist
⇒ requires non-malleability

[IKSS21]: realized using two-party computation, split-state non-malleable codes

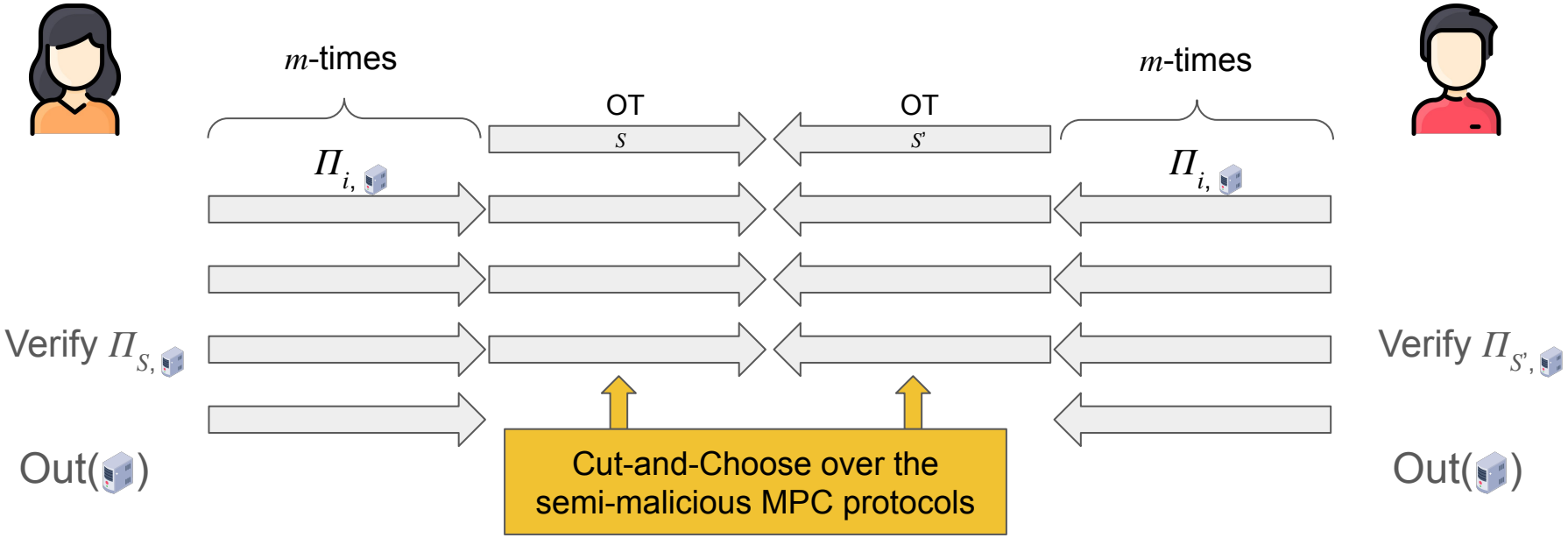
[IKSS21] Protocol

[IKSS21] Protocol

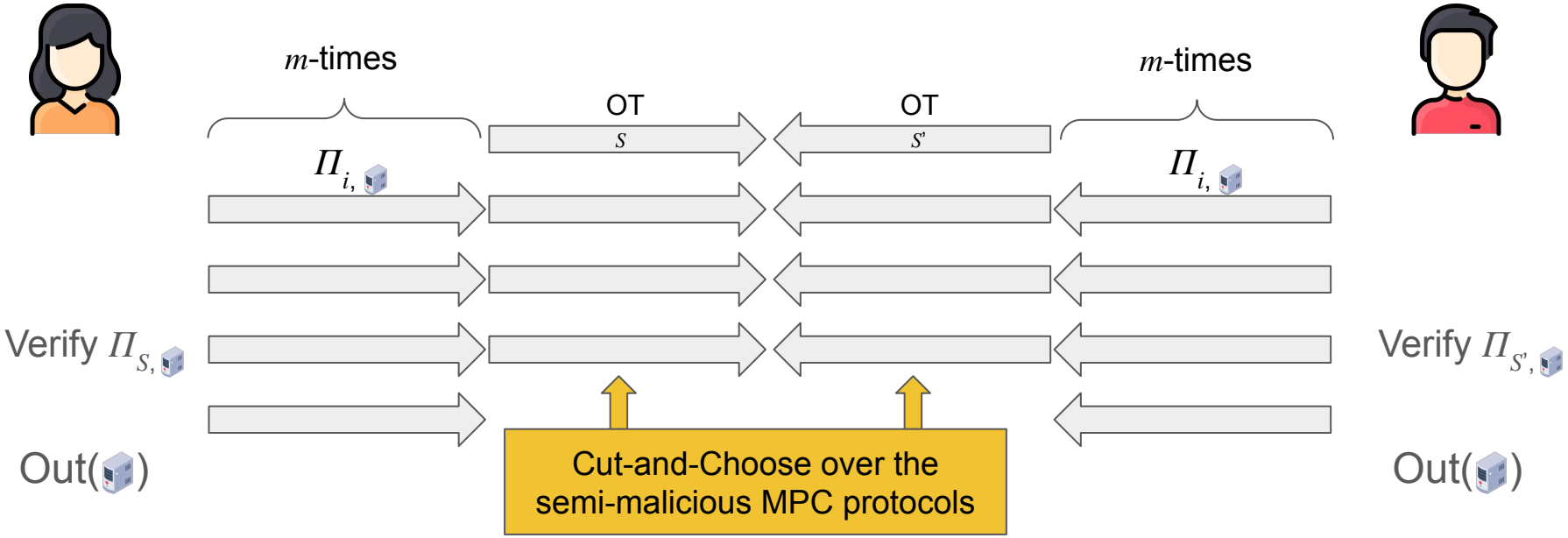
[IKSS21] Protocol



[IKSS21] Protocol

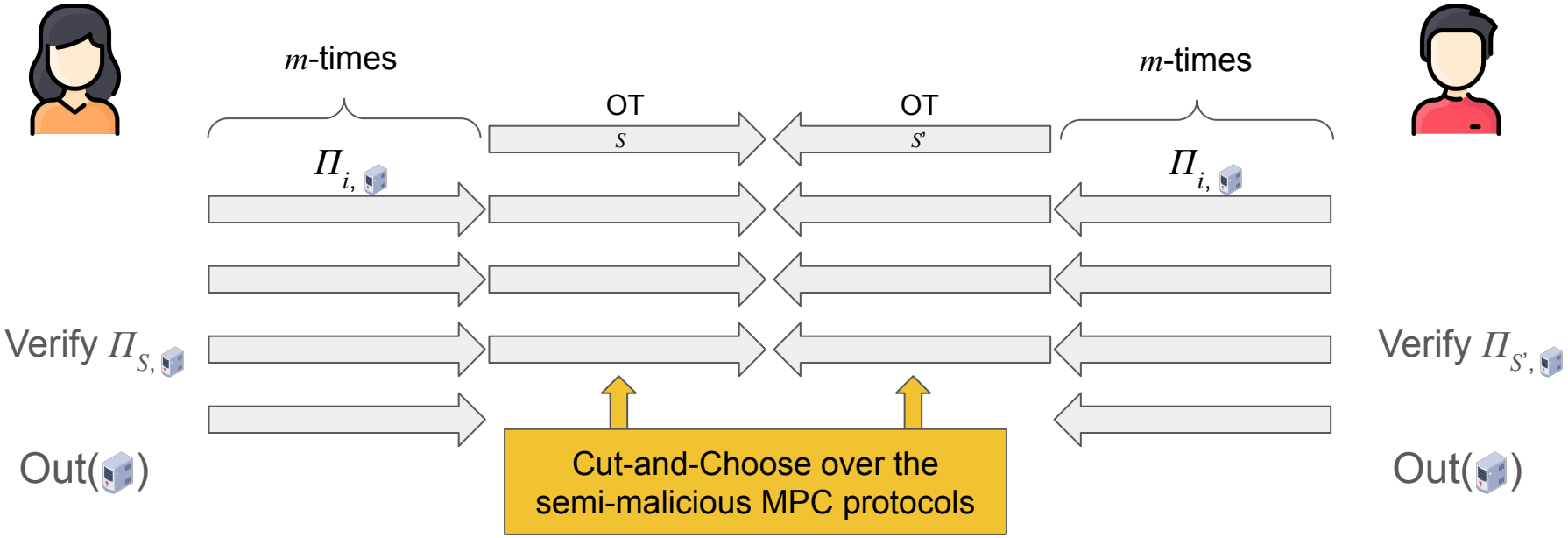


[IKSS21] Protocol



Bottleneck is the OT protocol

[IKSS21] Protocol



Bottleneck is the OT protocol →Can we do it in 3 rounds?

3 Round OT

No!

3 Round OT

No!

⇒ Impossibility of 4 round coin-tossing [KO04]

3 Round OT

No!

⇒ Impossibility of 4 round coin-tossing [KO04]

To circumvent the impossibility, we consider a relaxed OT security notion

3 Round OT

No!

⇒ Impossibility of 4 round coin-tossing [KO04]

To circumvent the impossibility, we consider a relaxed OT security notion

⇒ List OT based on List Coin-Tossing [BGJ+18]

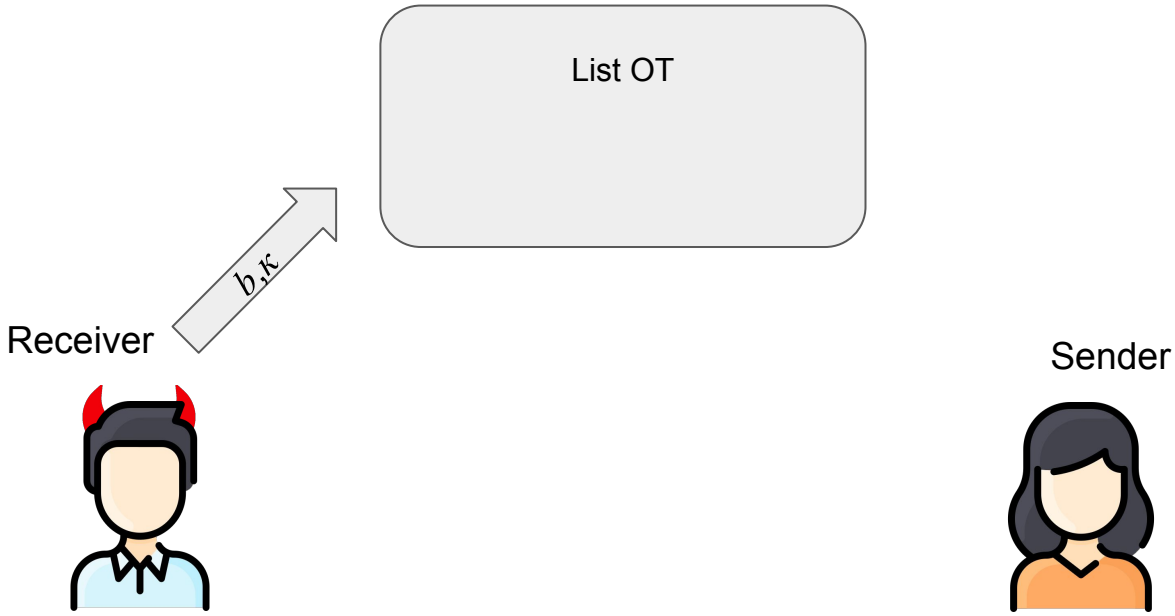
List OT

List OT

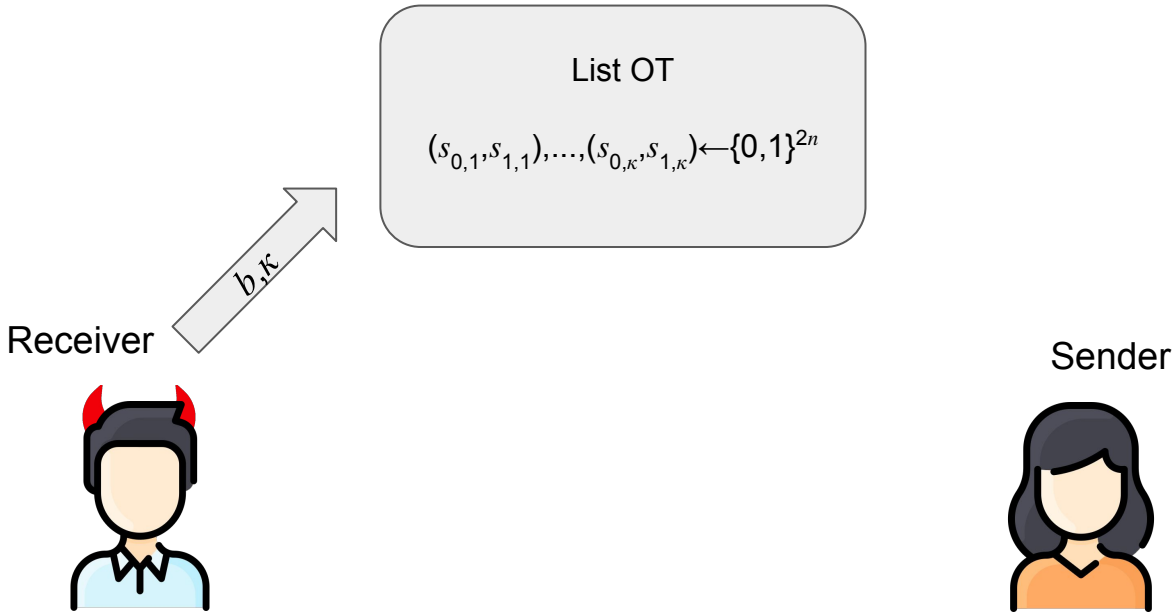
Receiver

Sender

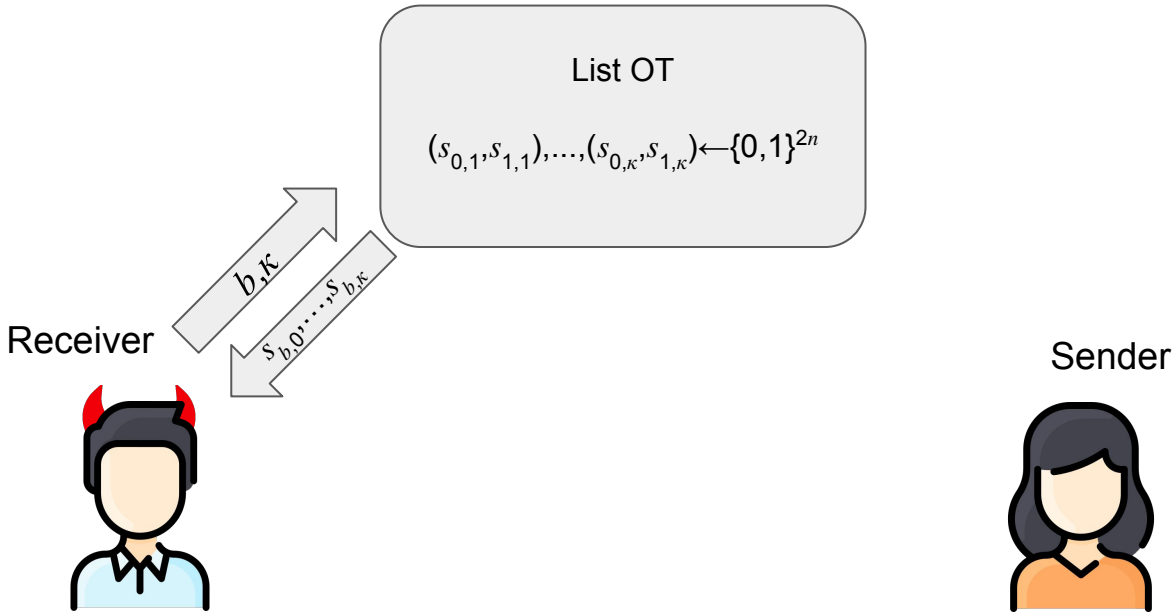
List OT



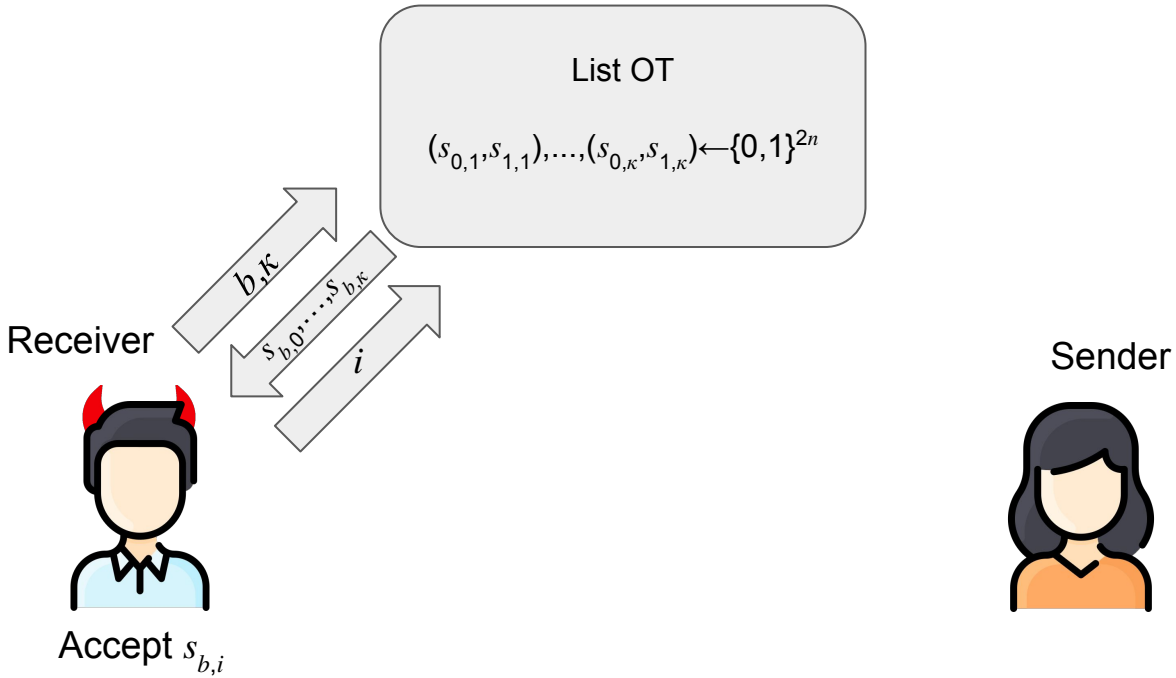
List OT


List OT
$(s_{0,1},s_{1,1}),\ldots,(s_{0,\kappa},s_{1,\kappa})\leftarrow\{0,1\}^{2n}$
b,κ
Receiver
Sender

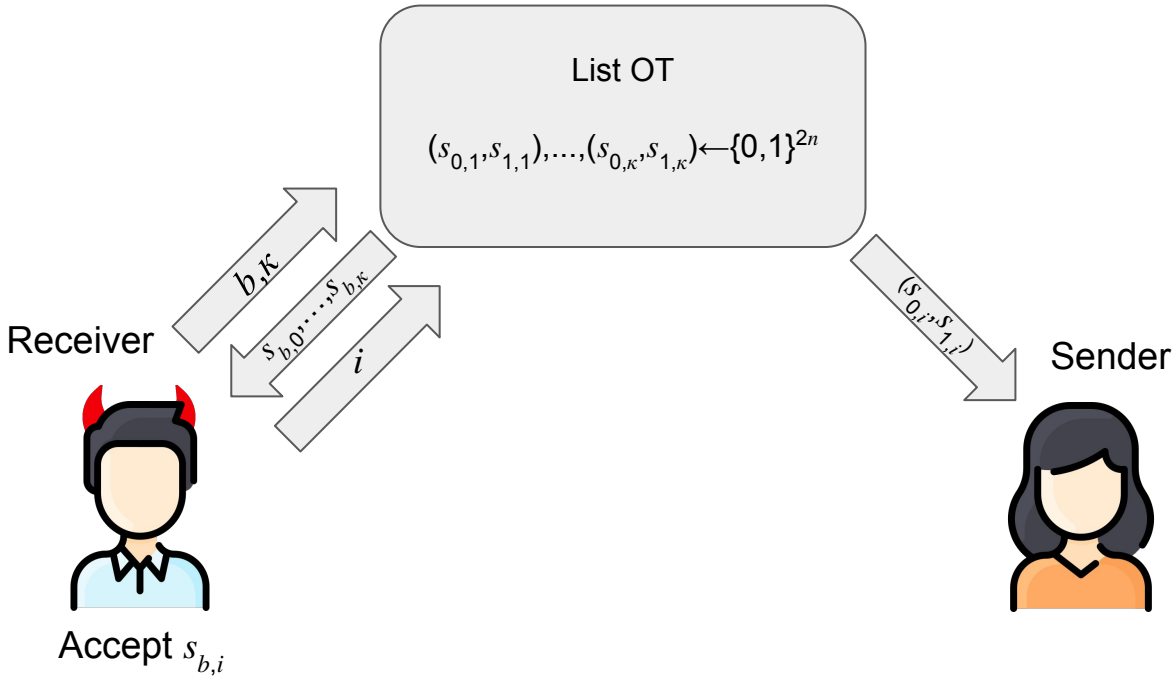
List OT


List OT
$(s_{0,1},s_{1,1}),\ldots,(s_{0,\kappa},s_{1,\kappa})\leftarrow\{0,1\}^{2n}$
b,κ
$s_{b,0},\ldots,s_{b,\kappa}$
Receiver
Sender

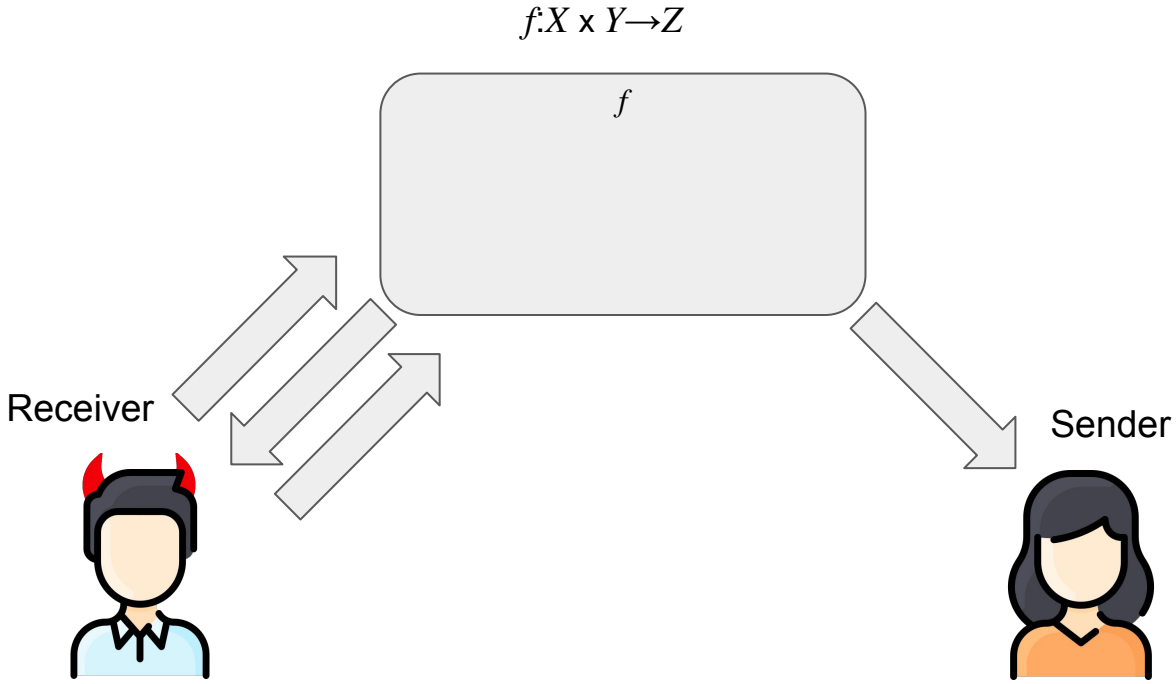
List OT


List OT
$(s_{0,1},s_{1,1}),\ldots,(s_{0,\kappa},s_{1,\kappa})\leftarrow\{0,1\}^{2n}$
b,κ
$s_{b,0},\ldots,s_{b,\kappa}$
i
Receiver
Sender
Accept $s_{b,i}$

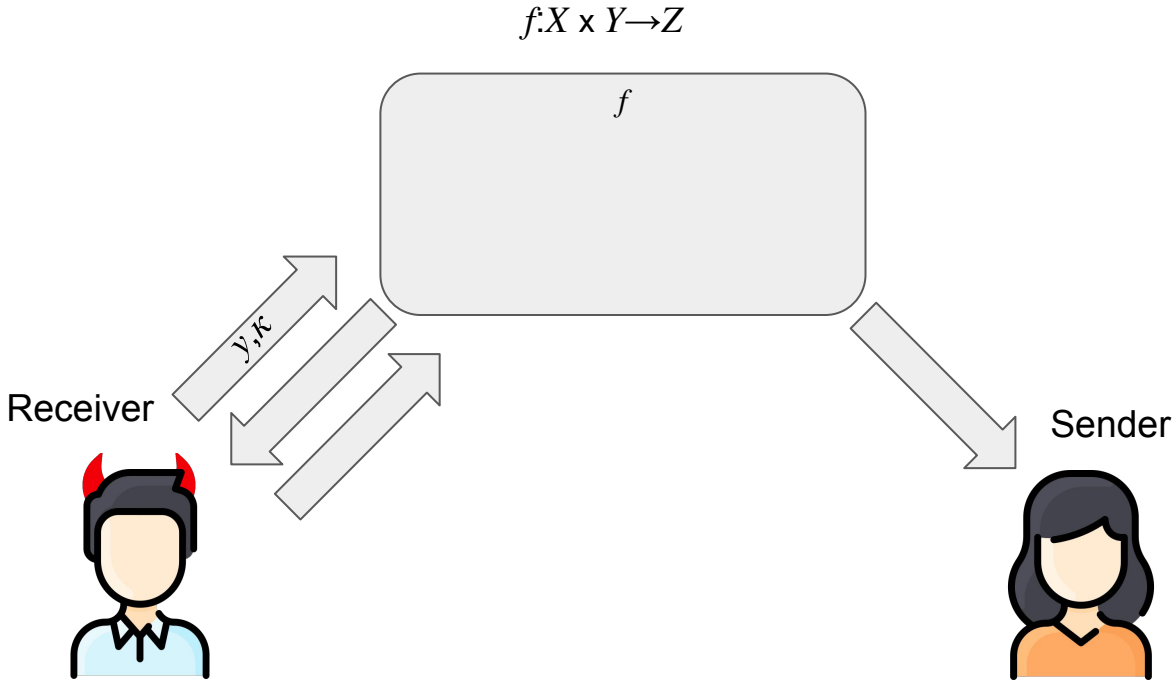
List OT


List OT
$(s_{0,1},s_{1,1}),\ldots,(s_{0,\kappa},s_{1,\kappa})\leftarrow\{0,1\}^{2n}$
b,κ
$s_{b,0},\ldots,s_{b,\kappa}$
i
$(s_{0,i},s_{1,i})$
Receiver
Sender
Accept $s_{b,i}$

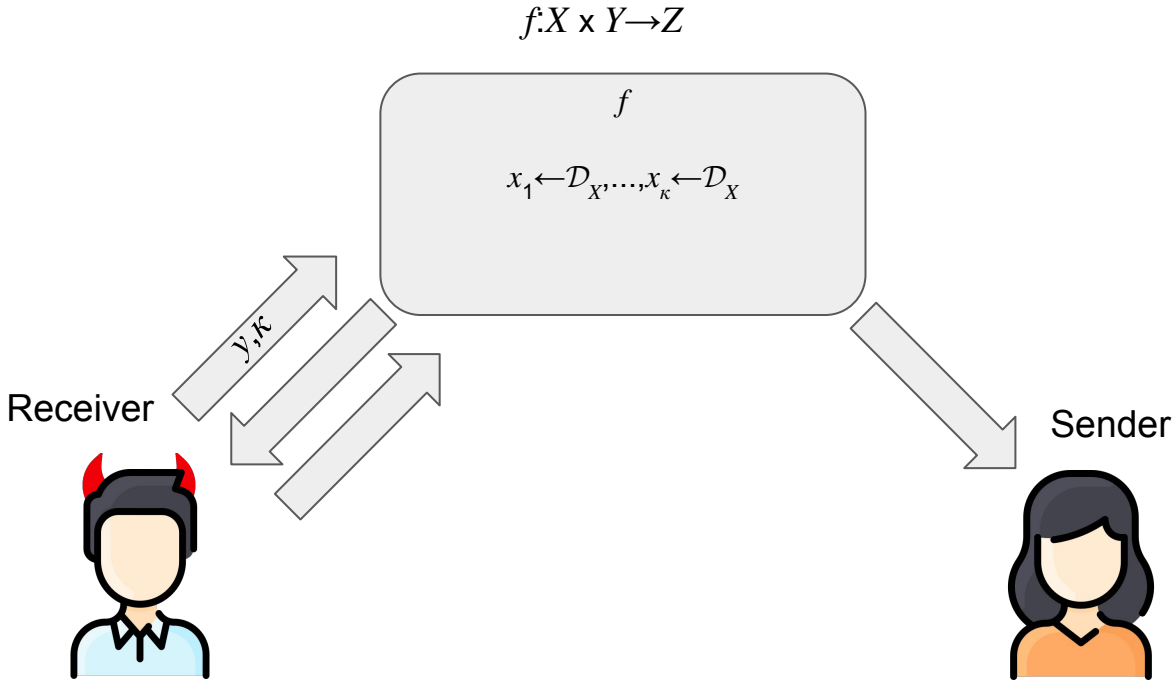
List Two-Party Computation



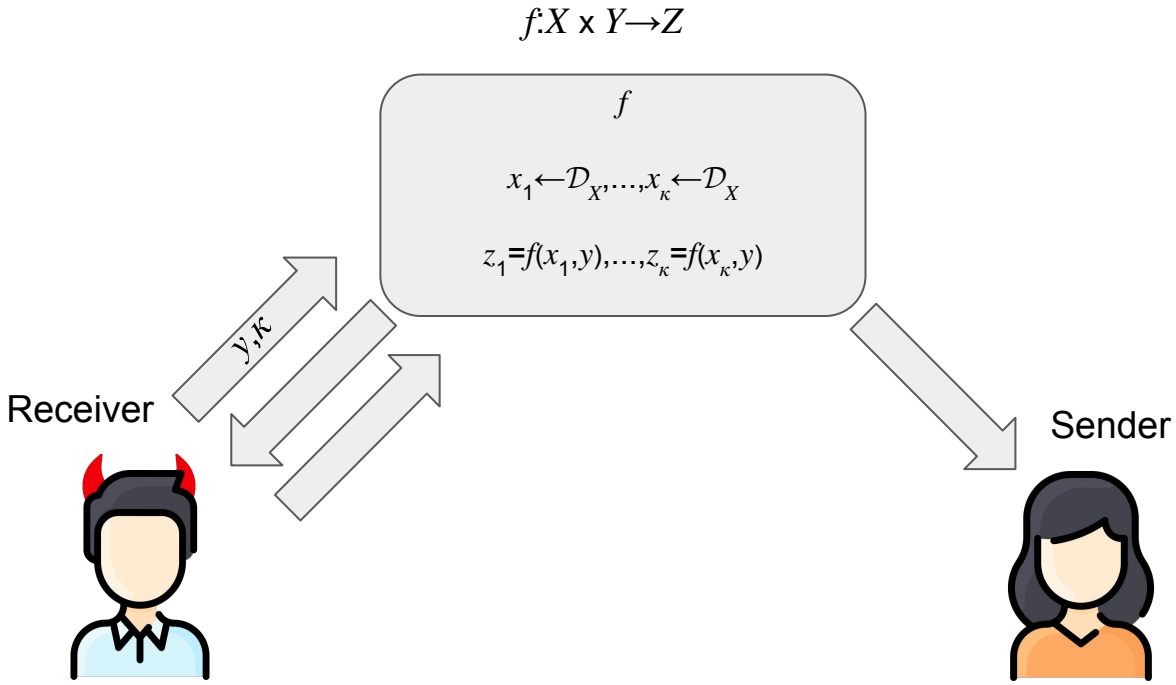
List Two-Party Computation



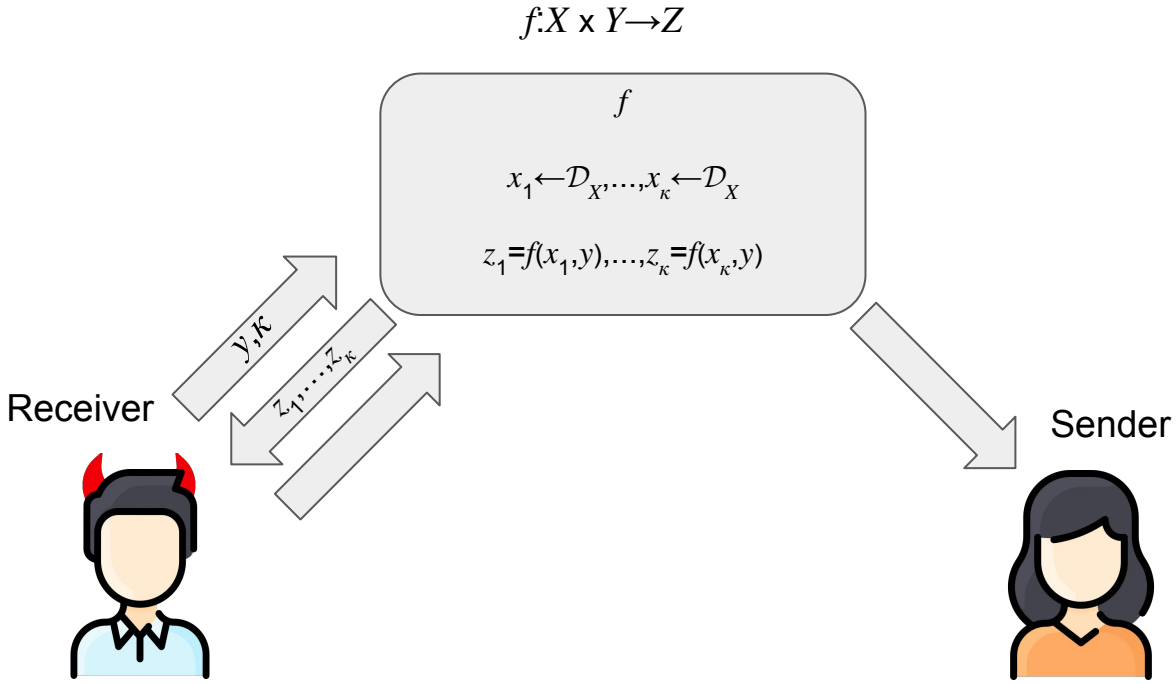
List Two-Party Computation



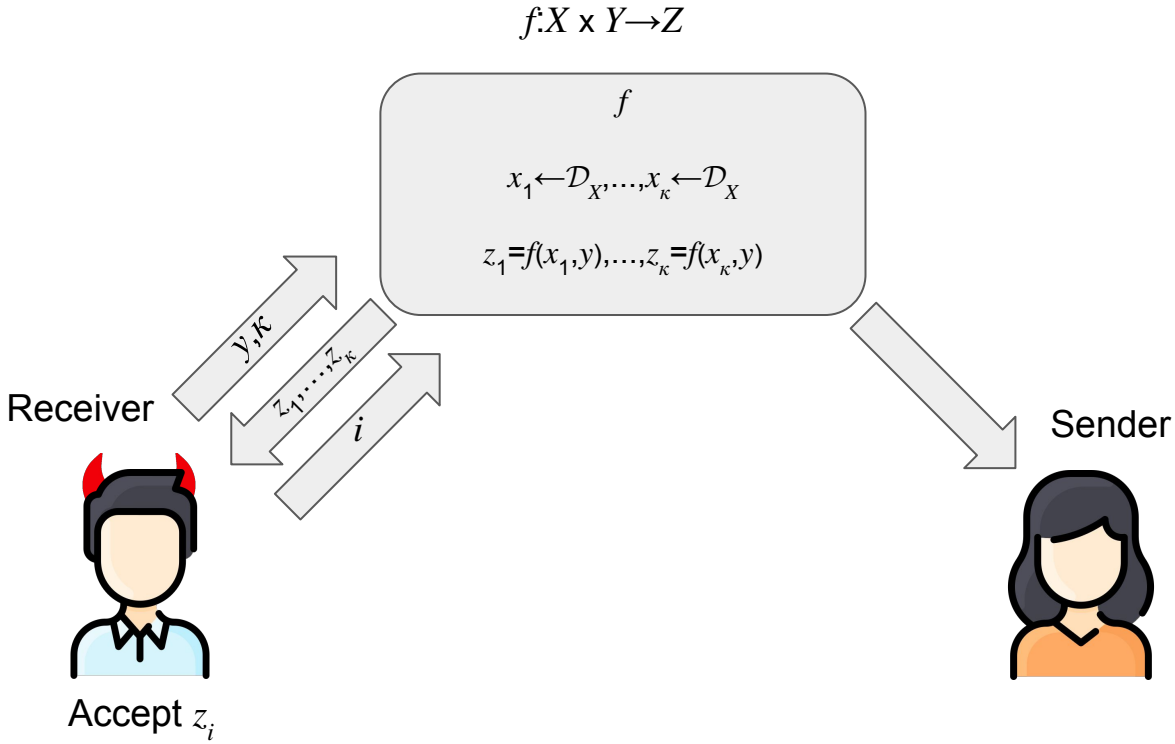
List Two-Party Computation



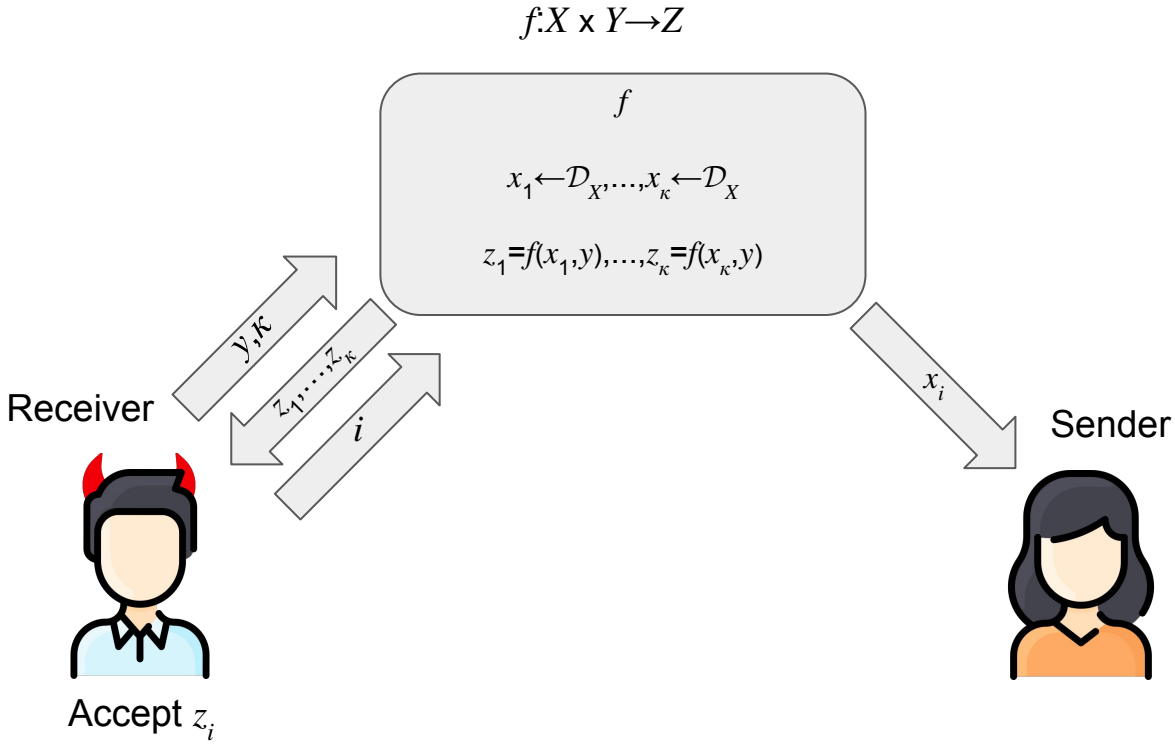
List Two-Party Computation



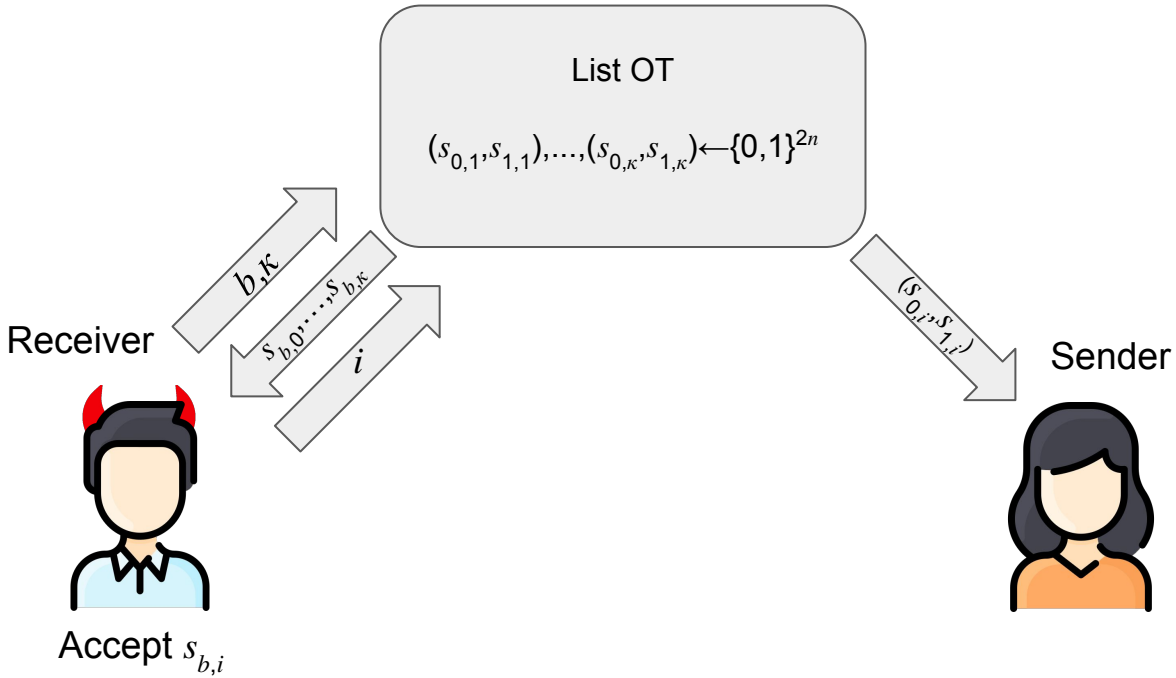
List Two-Party Computation



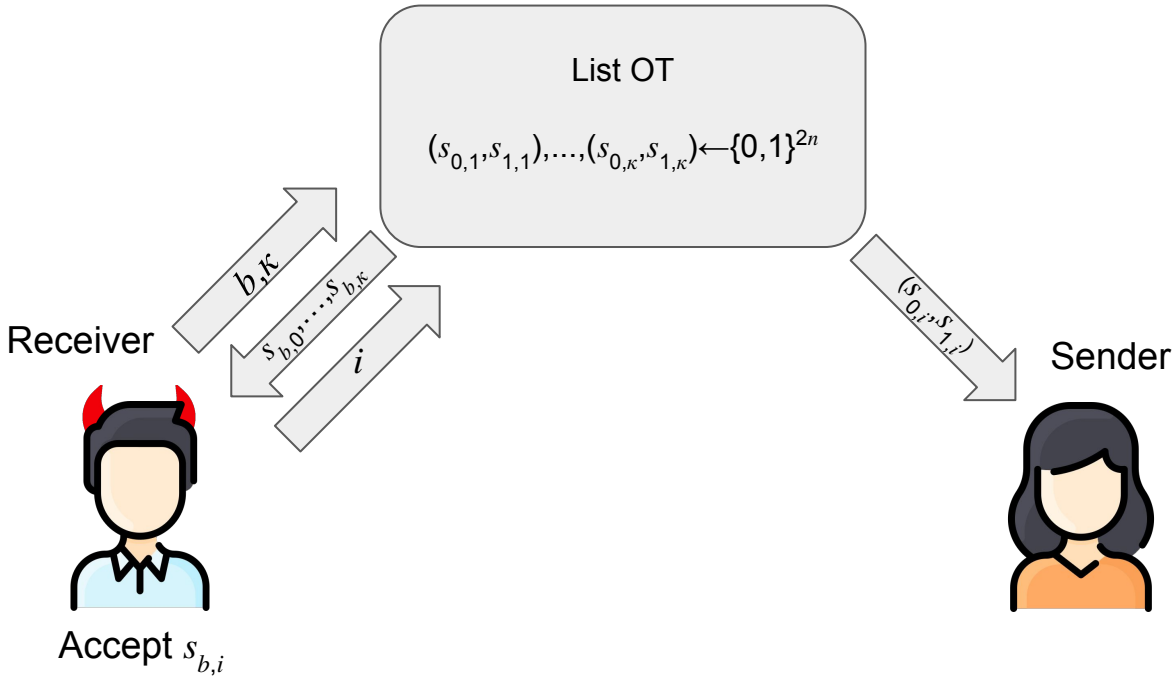
List Two-Party Computation



List OT

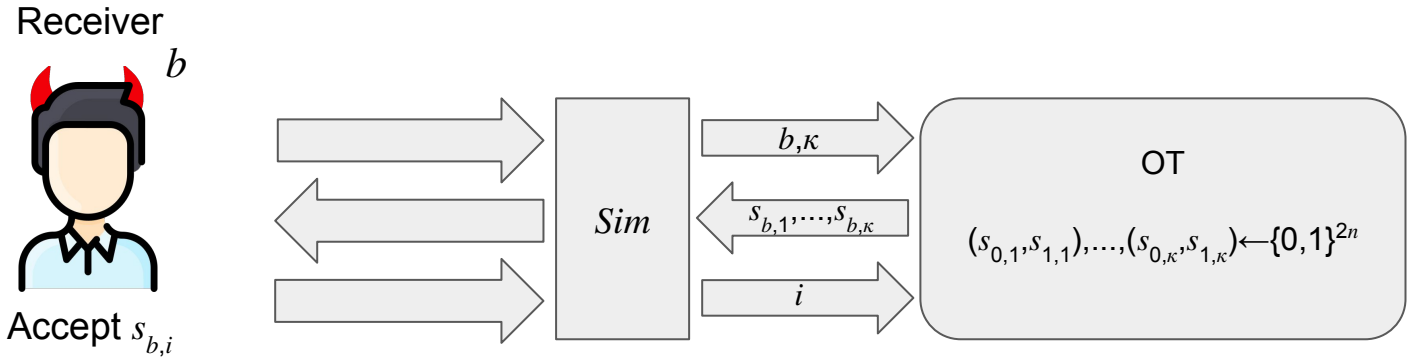

List OT
$(s_{0,1},s_{1,1}),\ldots,(s_{0,\kappa},s_{1,\kappa})\leftarrow\{0,1\}^{2n}$
b,κ
$s_{b,0},\ldots,s_{b,\kappa}$
i
$(s_{0,i},s_{1,i})$
Receiver
Sender
Accept $s_{b,i}$

List OT

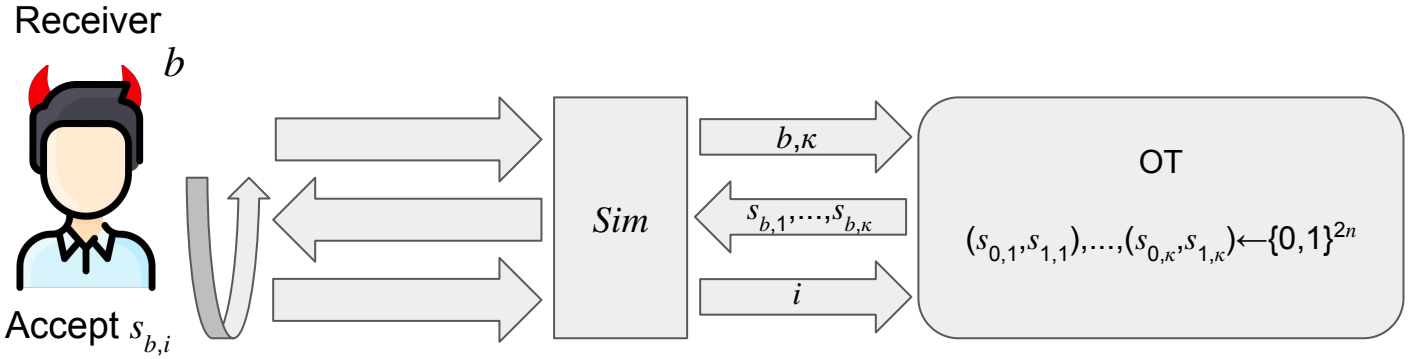


How to do it in 3 rounds?/How to simulate?

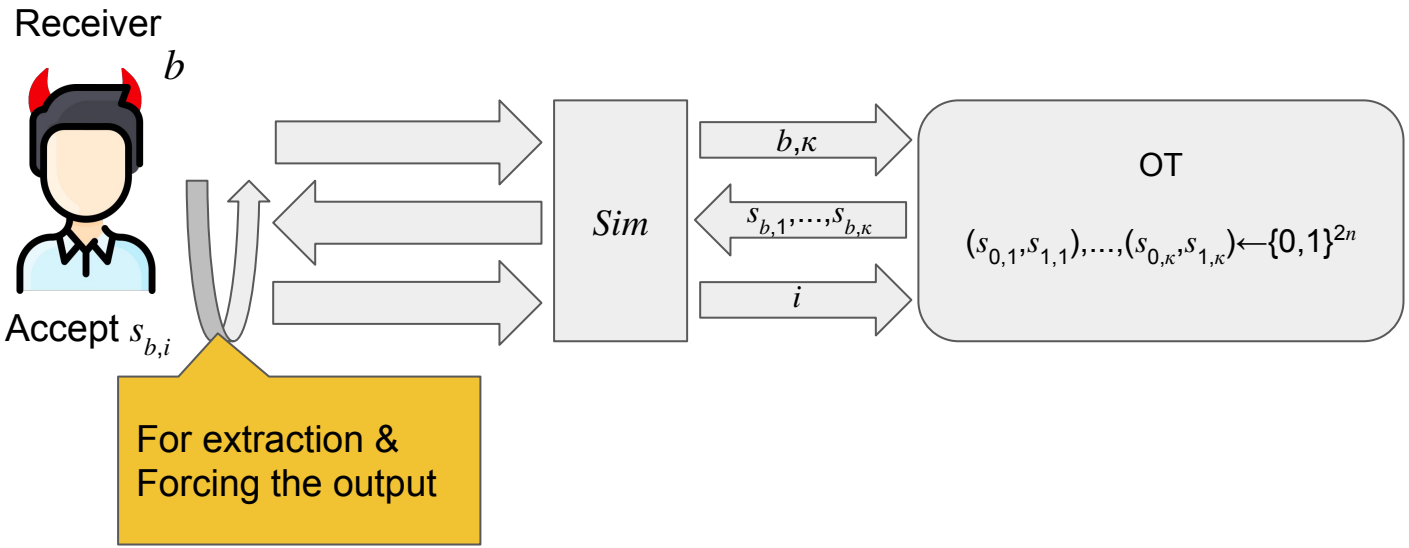
List OT Simulation



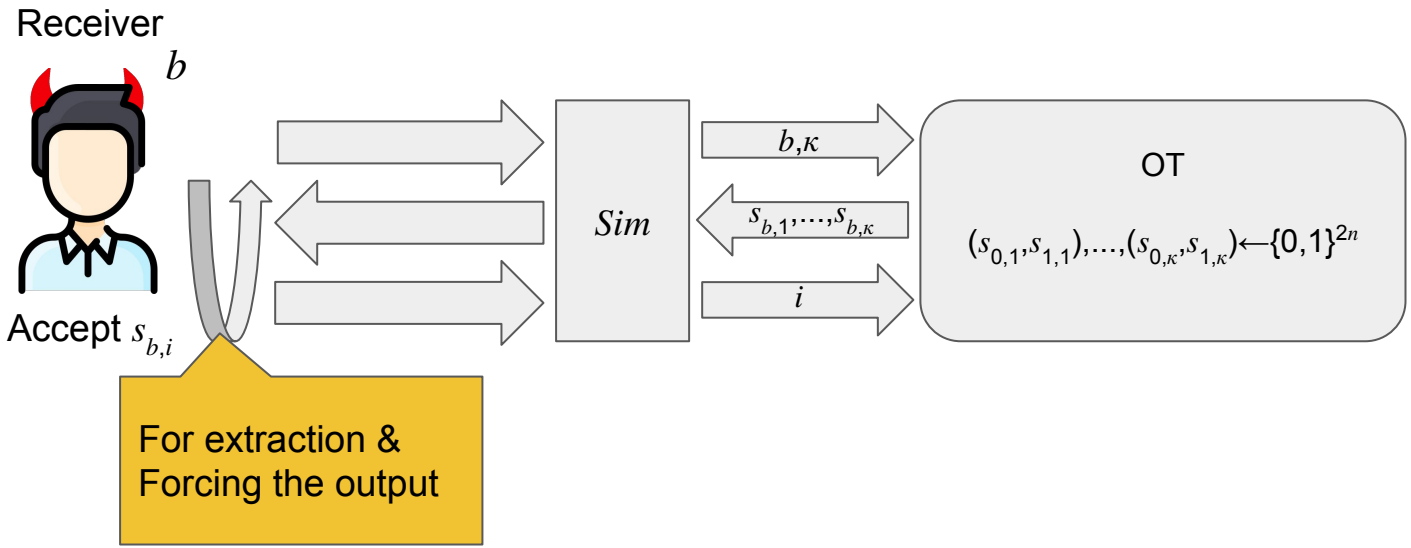
List OT Simulation



List OT Simulation



List OT Simulation



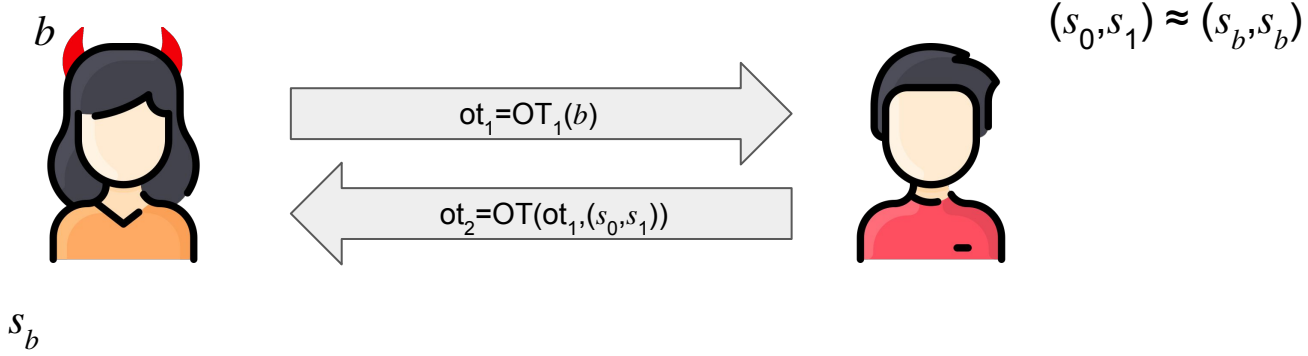
How to construct something like this?

List Oblivious Transfer

We start from a sender private OT

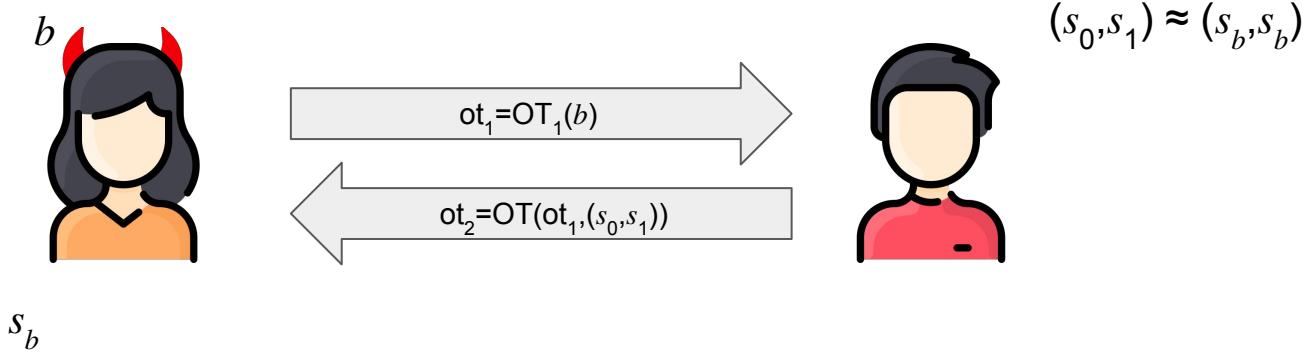
List Oblivious Transfer

We start from a sender private OT



List Oblivious Transfer

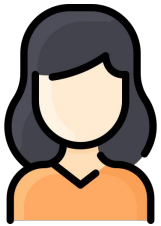
We start from a sender private OT



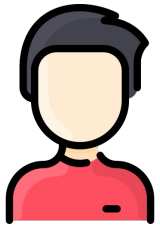
Cut-and-Choose using (k,n) OT combiners [HKN+05]

List Oblivious Transfer

$b=b_0 \oplus b_1$

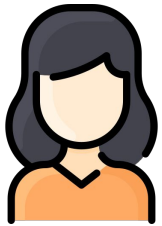


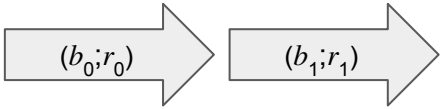
(s_0, s_1)

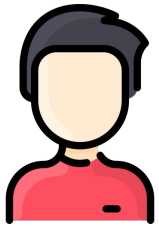


List Oblivious Transfer

$b=b_0 \oplus b_1$



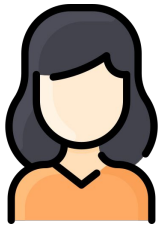




(s_0,s_1)

List Oblivious Transfer

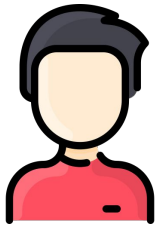
$b=b_0 \oplus b_1$



$(b_0;r_0)$

$(b_1;r_1)$

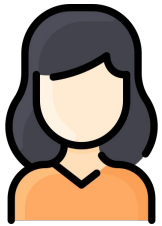
(s_0,s_1)

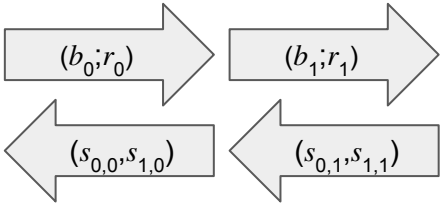


$s_{0,0}, s_{1,0}, s_{0,1}, s_{1,1} = \mathsf{Share}(s_0, s_1)$

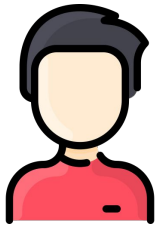
List Oblivious Transfer

$b=b_0 \oplus b_1$



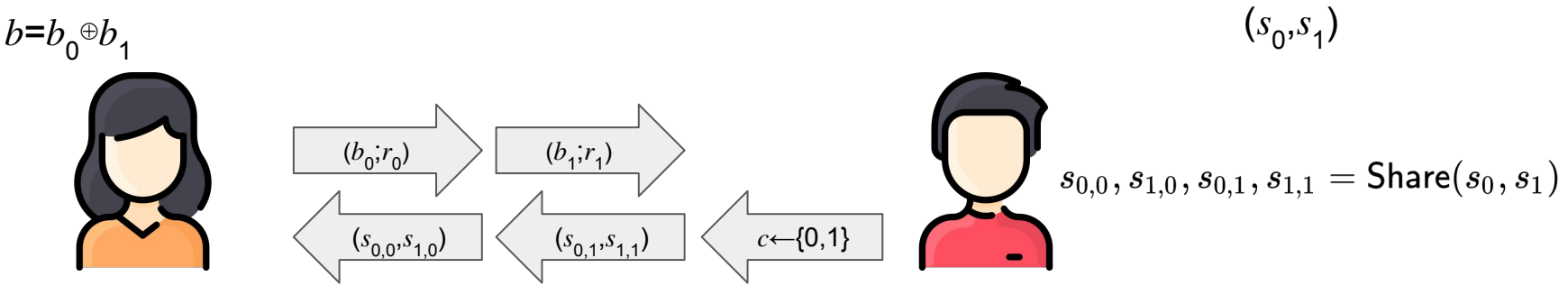


(s_0,s_1)

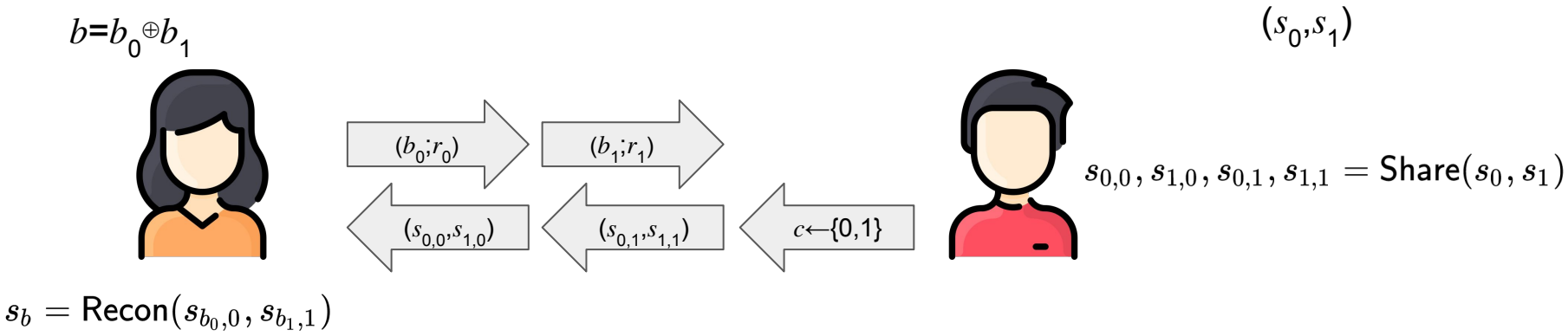


$s_{0,0}, s_{1,0}, s_{0,1}, s_{1,1} = \mathsf{Share}(s_0, s_1)$

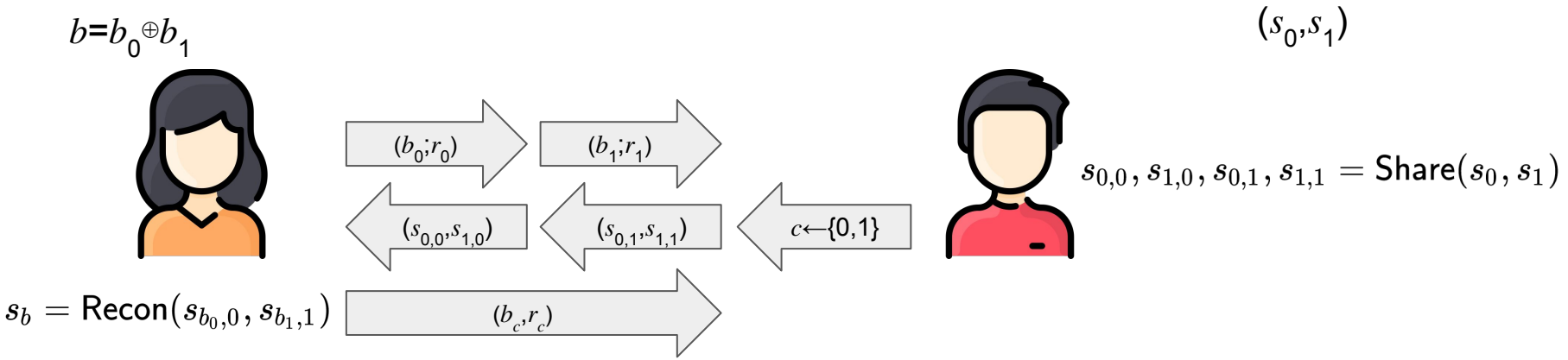
List Oblivious Transfer



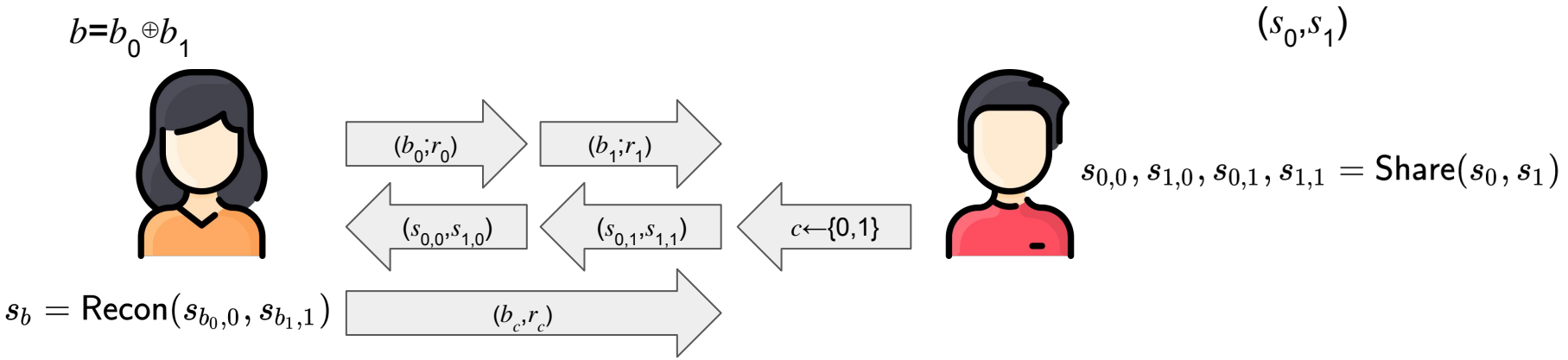
List Oblivious Transfer



List Oblivious Transfer

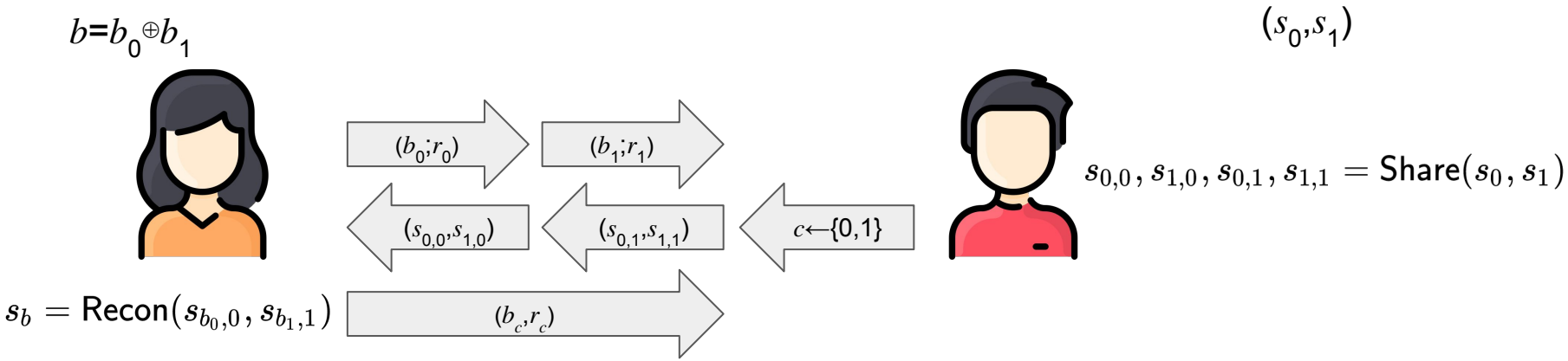


List Oblivious Transfer



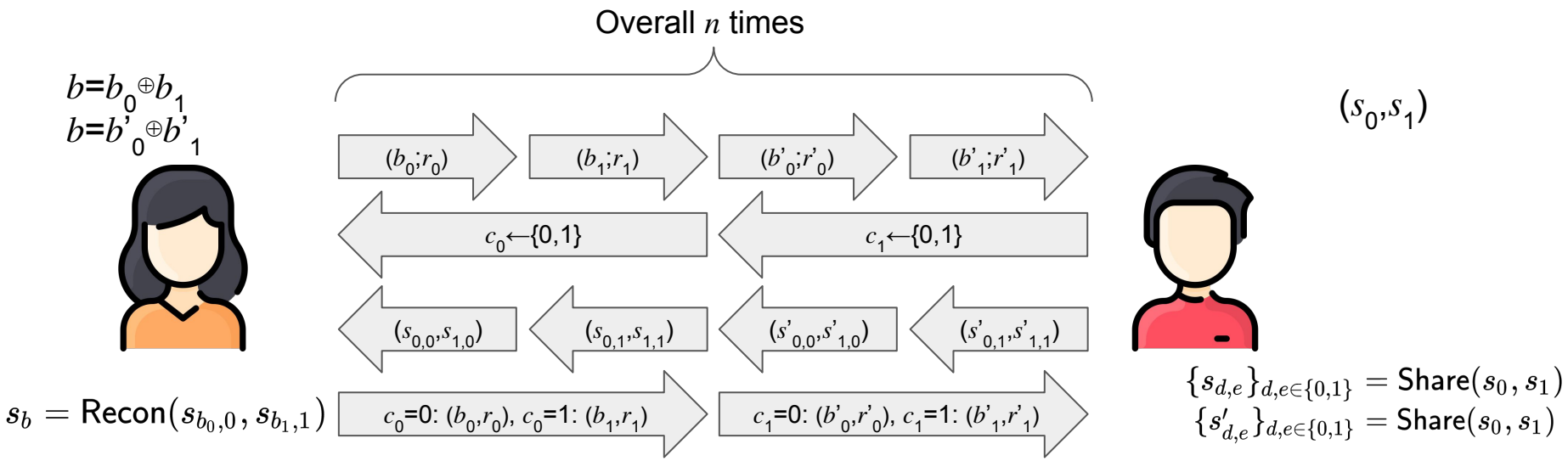
Still not possible to extract since simulator needs all positions opened

List Oblivious Transfer

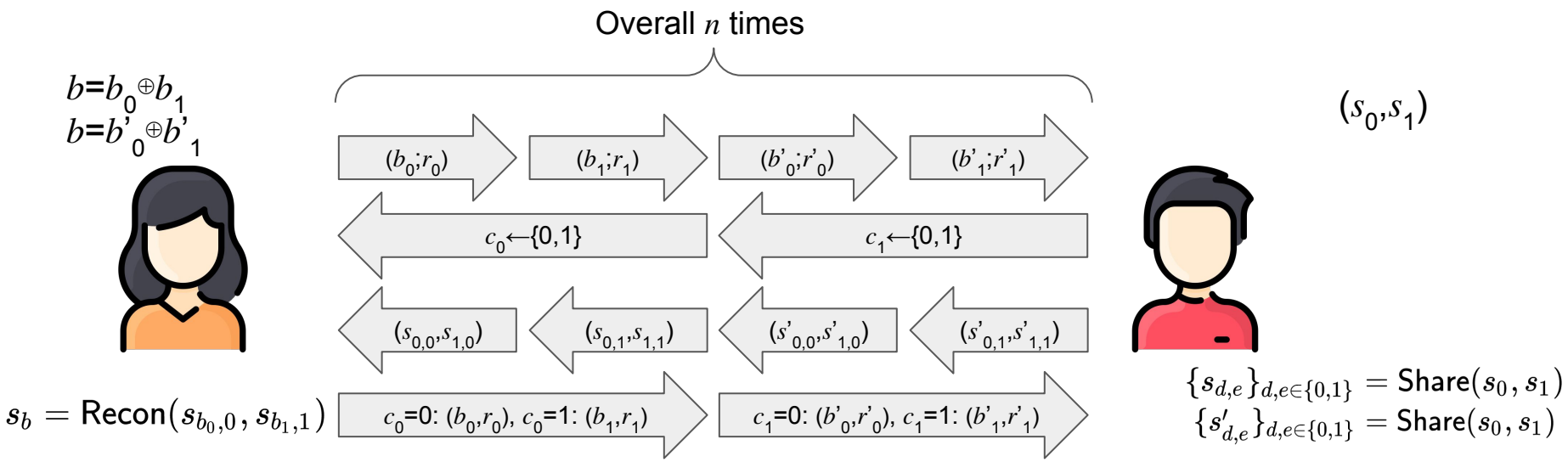


Still not possible to extract since simulator needs all positions opened→Repeat n times

List Oblivious Transfer

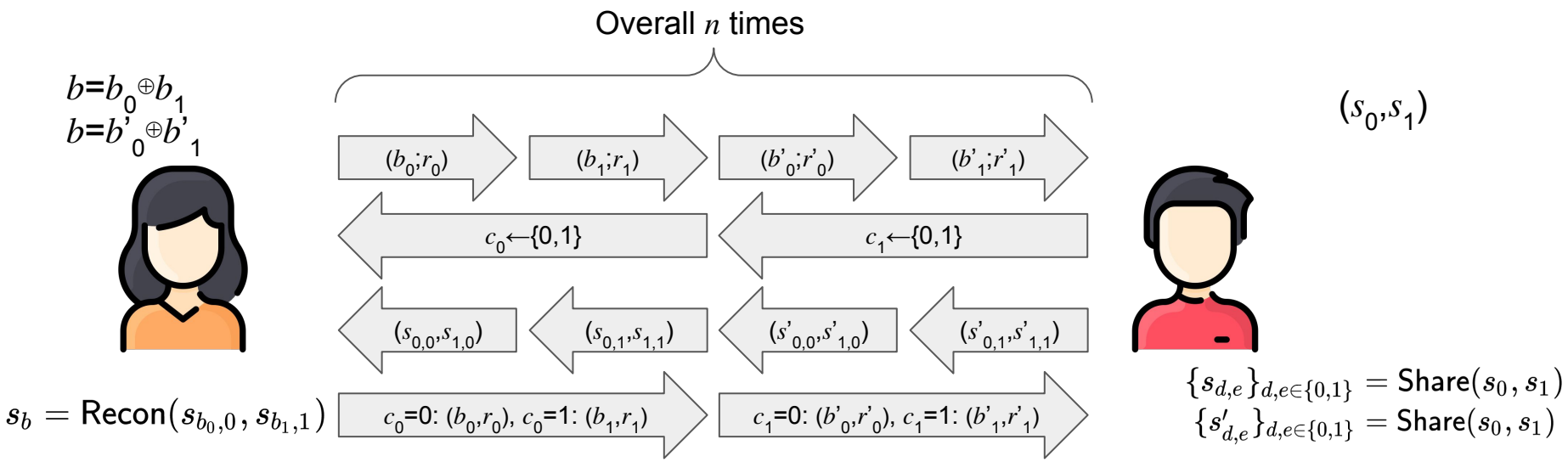


List Oblivious Transfer



Adversary can use different inputs in different instances!

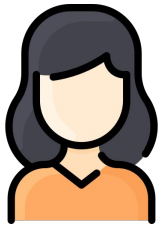
List Oblivious Transfer



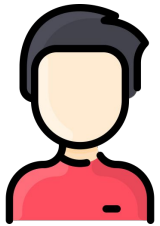
Adversary can use different inputs in different instances!
→ We need to connect all the executions to a single input

List Oblivious Transfer

b

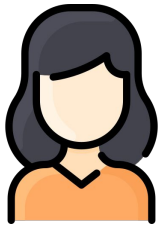


(s_0, s_1)

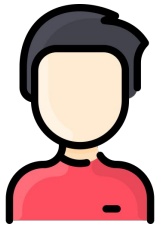


List Oblivious Transfer

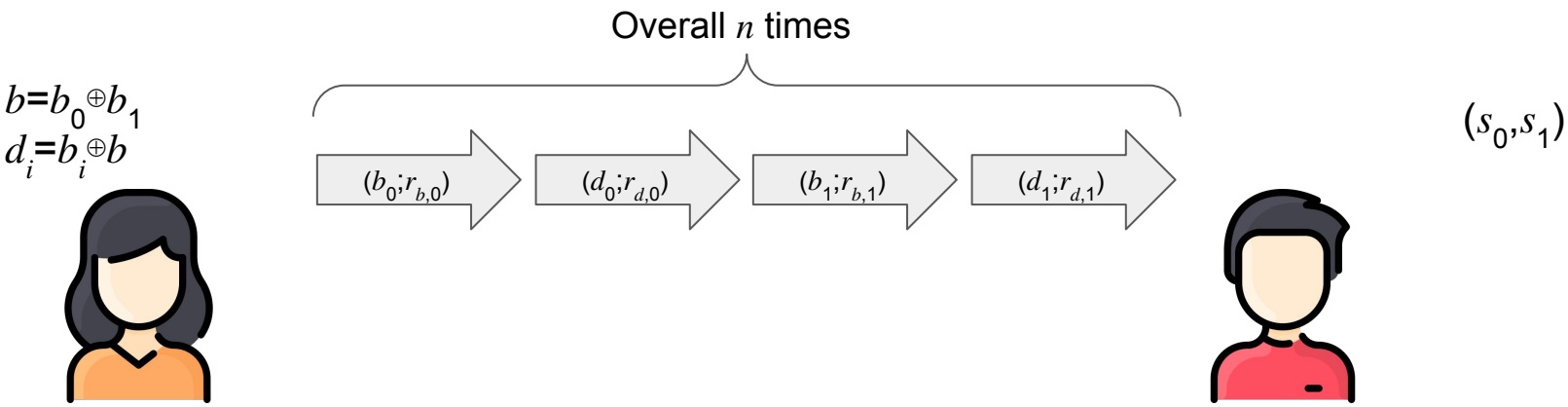
$b=b_0 \oplus b_1$
$d_i=b_i \oplus b$



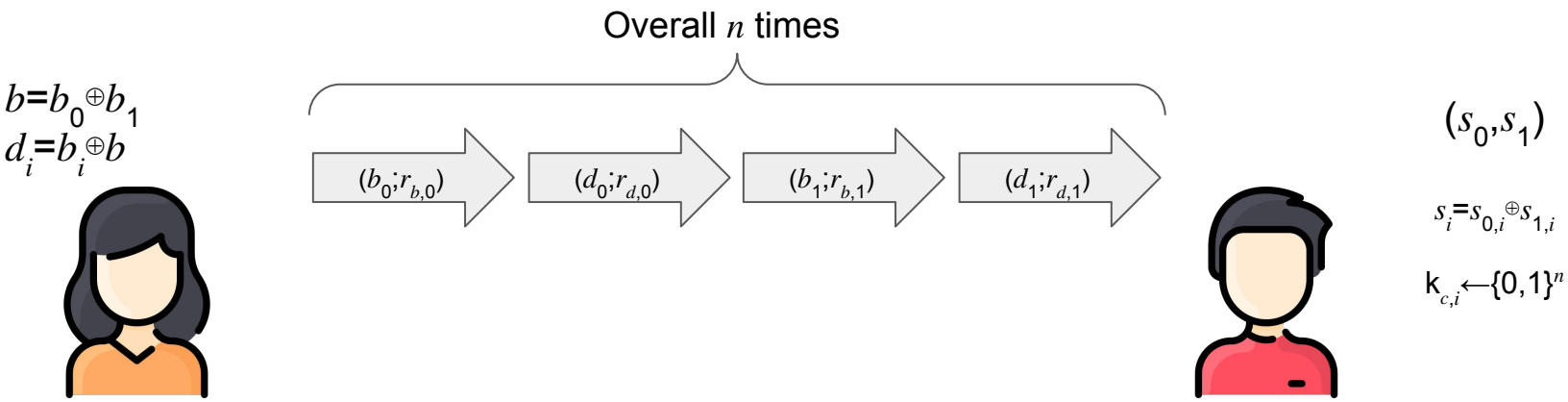
(s_0,s_1)



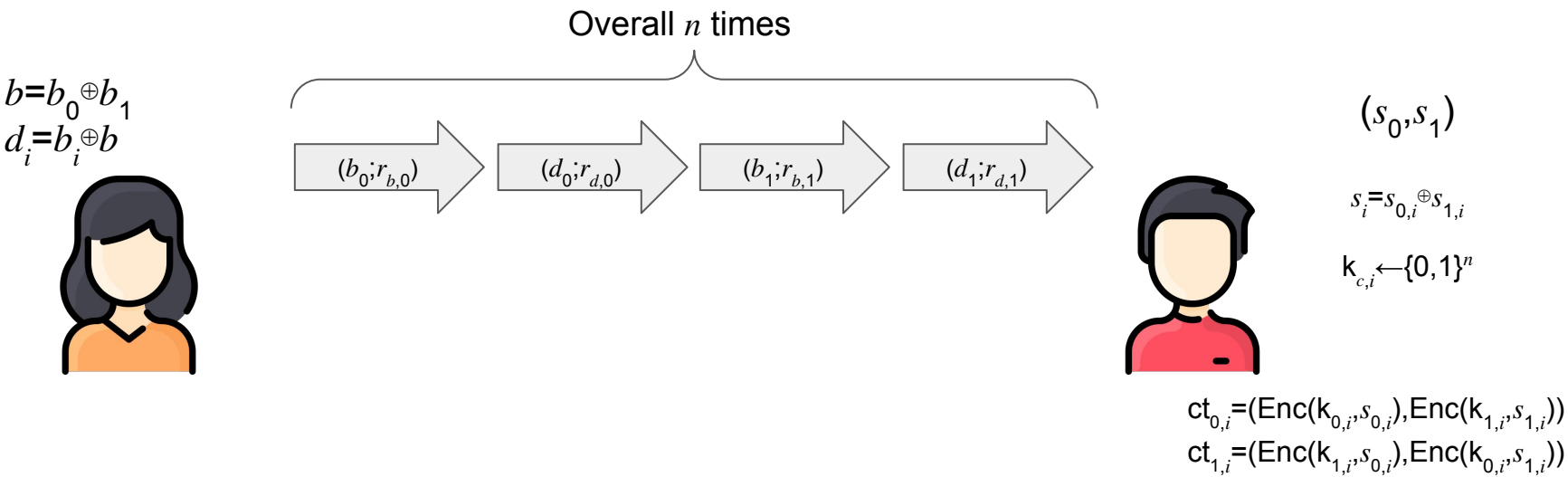
List Oblivious Transfer



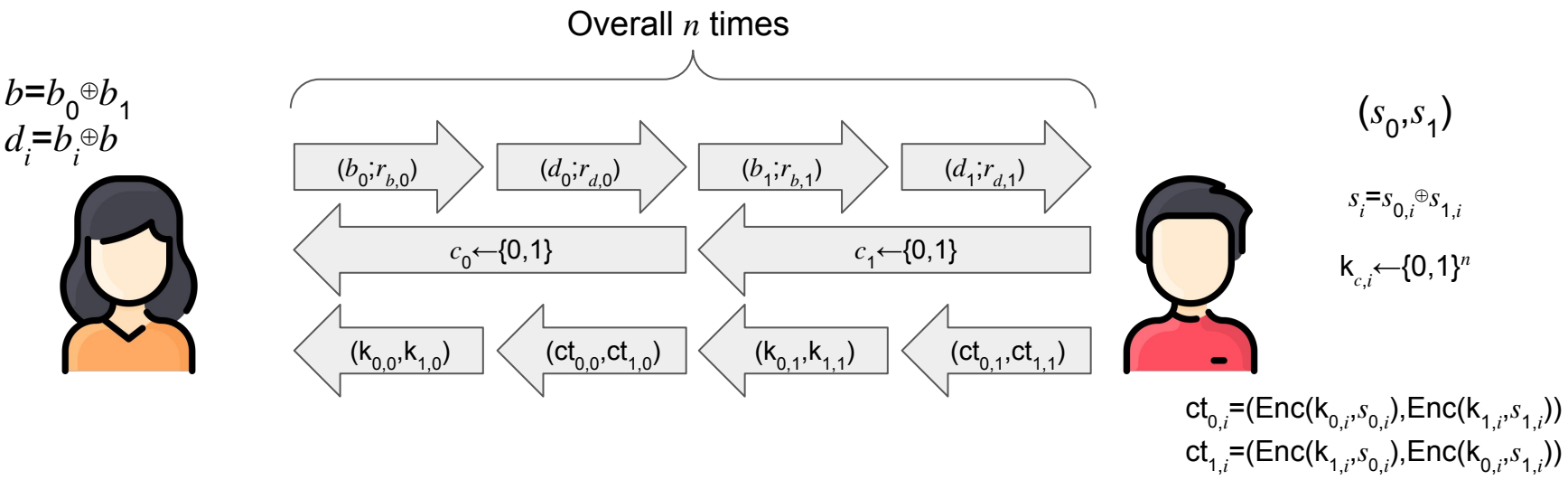
List Oblivious Transfer



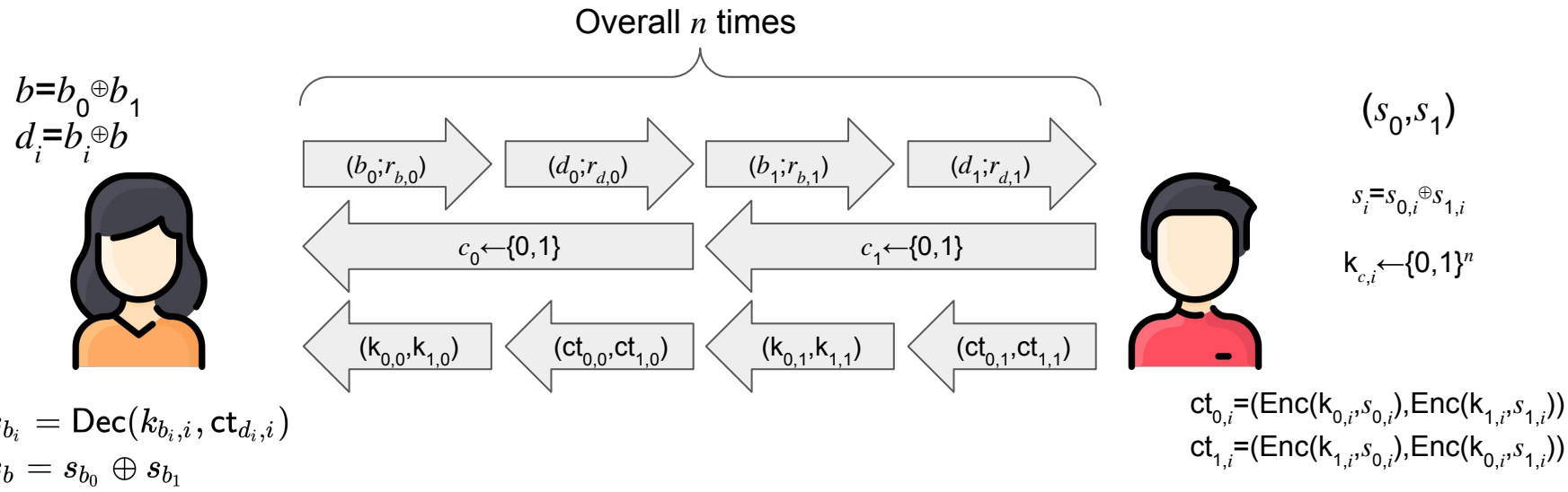
List Oblivious Transfer



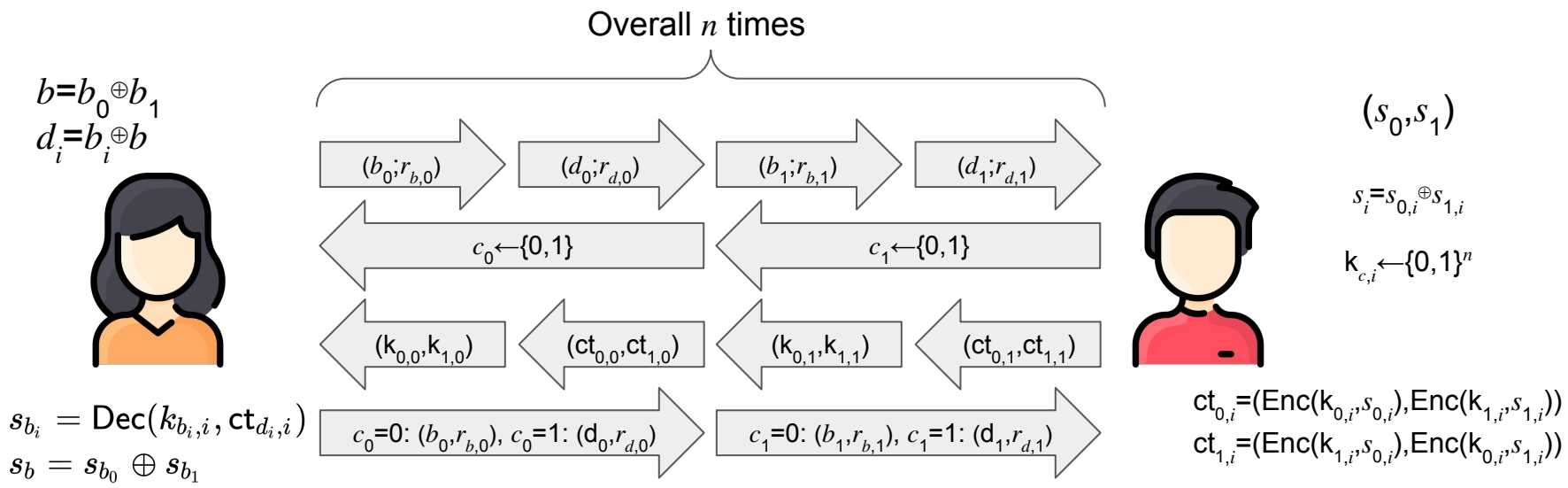
List Oblivious Transfer



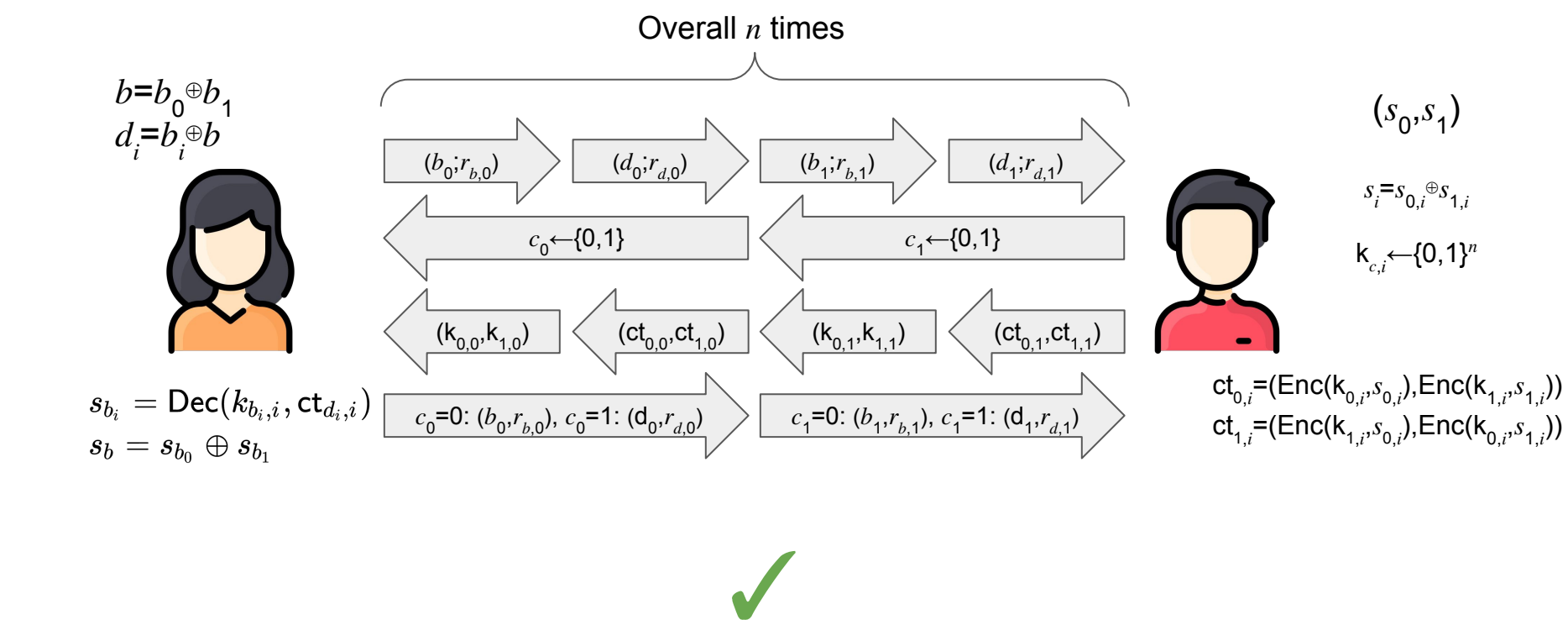
List Oblivious Transfer



List Oblivious Transfer



List Oblivious Transfer

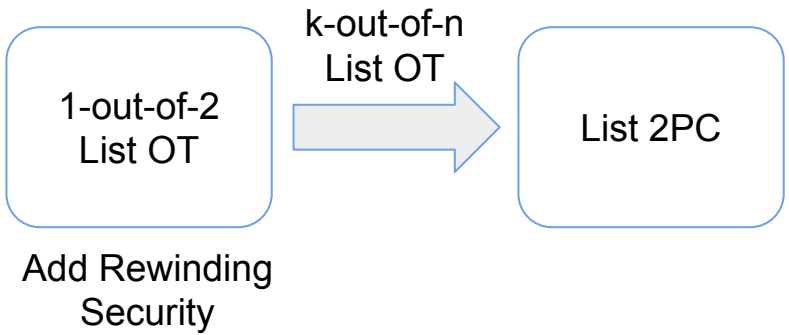


✓

Roadmap to Four-Round Multiparty Computation

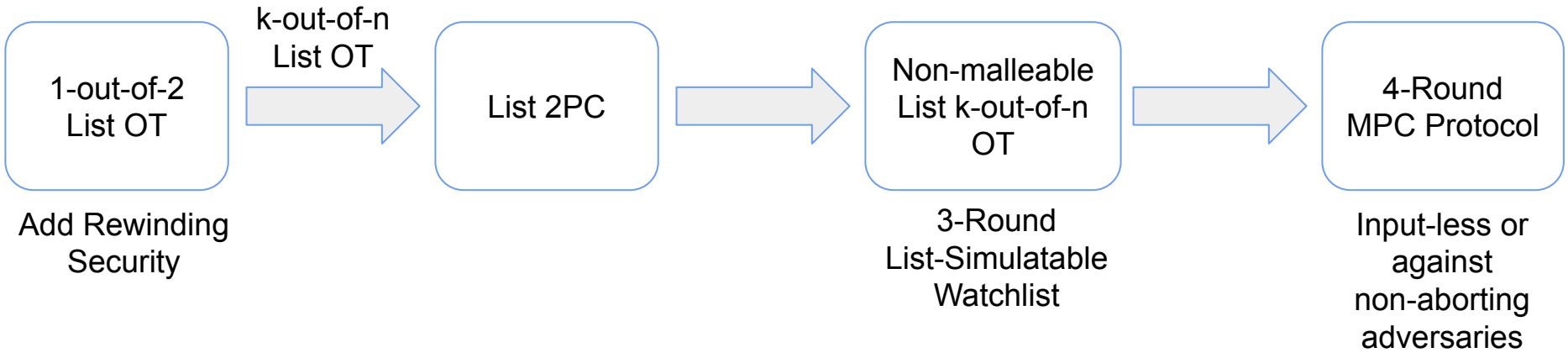
1-out-of-2
List OT

Roadmap to Four-Round Multiparty Computation

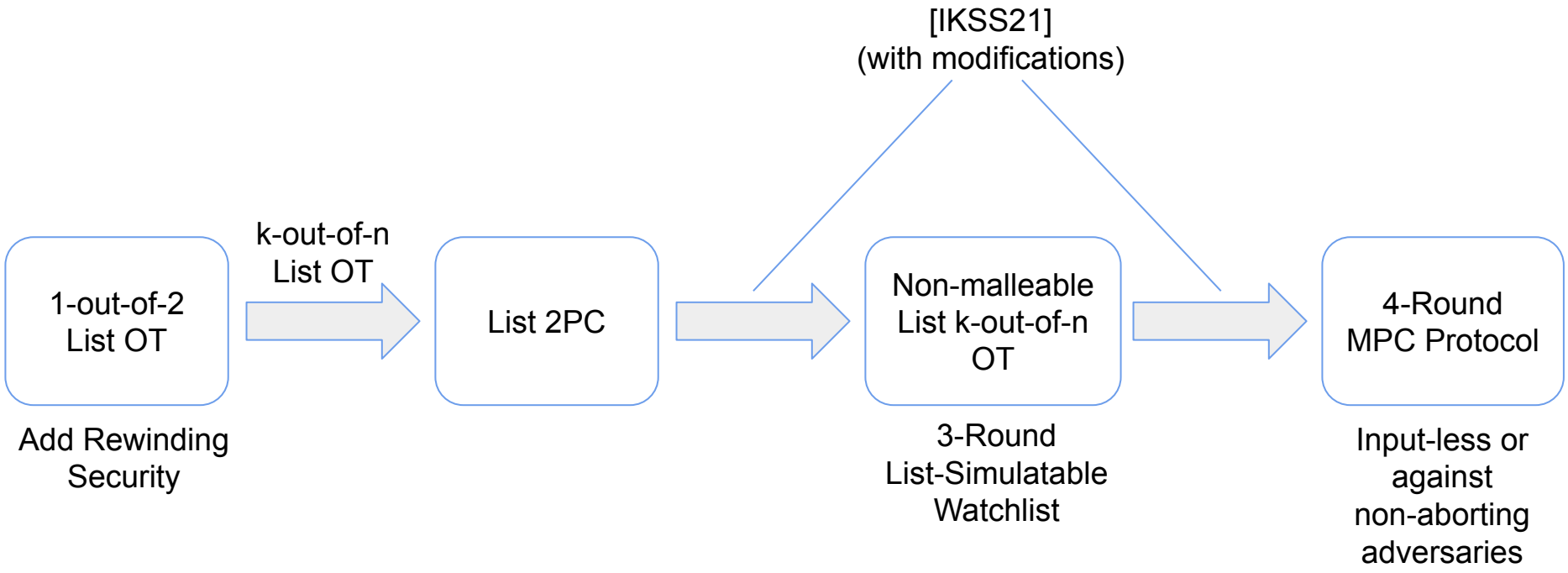


Roadmap to Four-Round Multiparty Computation

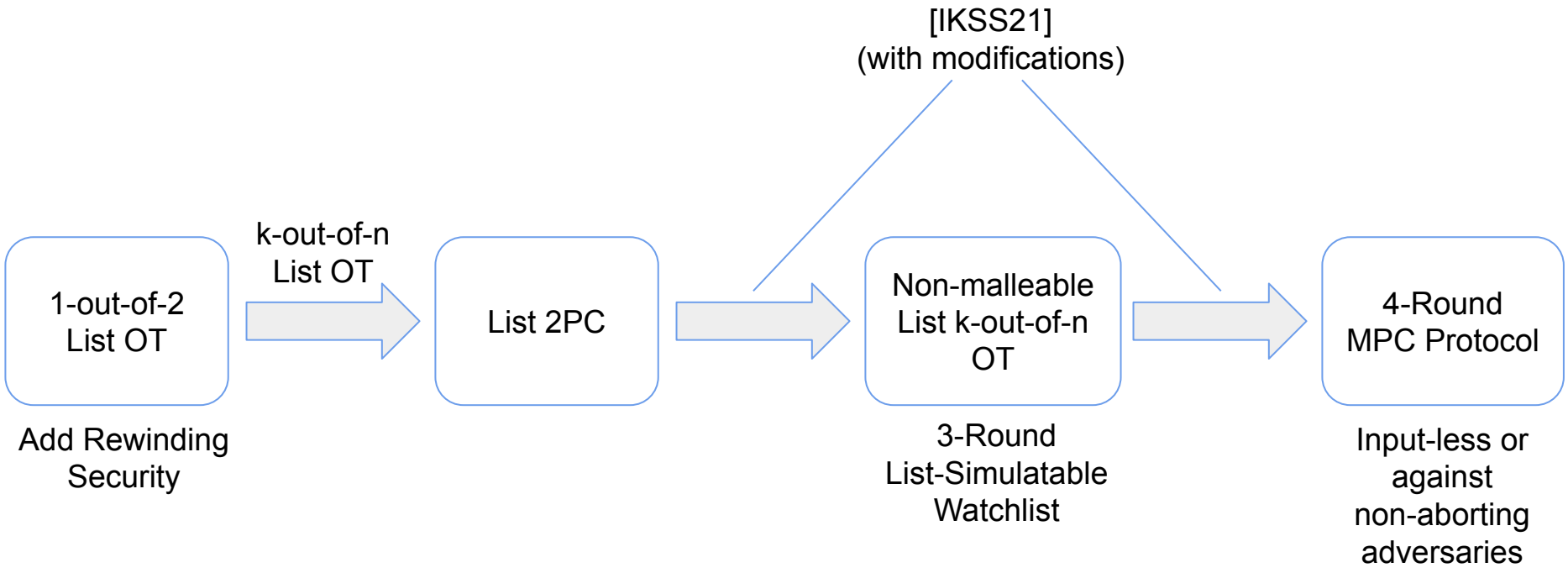
Roadmap to Four-Round Multiparty Computation



Roadmap to Four-Round Multiparty Computation



Roadmap to Four-Round Multiparty Computation



Thank You!

References

[BGJ+18] S. Badrinarayanan, V. Goyal, A. Jain, Y. T. Kalai, D. Khurana, and A. Sahai. Promise zero knowledge and its applications to round optimal MPC. In CRYPTO 2018.

[CCG+20] A. R. Choudhuri, M. Ciampi, V. Goyal, A. Jain, and R. Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In TCC 2020.

[GMPP16] S. Garg, P. Mukherjee, O. Pandey, and A. Polychroniadou. The exact round complexity of secure computation. In Eurocrypt 2016.

[HHPV18] S. Halevi, C. Hazay, A. Polychroniadou, and M. Venkitasubramaniam. Round-optimal secure multi-party computation. In CRYPTO 2018.

[HKN+05] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In EUROCRYPT 2005.

References

[IKSS21] Y. Ishai, D. Khurana, A. Sahai, and A. Srinivasan. On the round complexity of black-box secure MPC. In CRYPTO 2021.

[IKSS23] Y. Ishai, D. Khurana, A. Sahai, and A. Srinivasan. Round-optimal black-box MPC in the plain model. In CRYPTO 2023.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In CRYPTO 2008.

[KO04] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In CRYPTO 2004.