

EUROCRYPT 2026

Non-Interactive Secure Computation with Constant Communication Overhead

Yuval Ishai¹, **Ziyang Jin**², Naty Peter, Akshayaram Srinivasan²

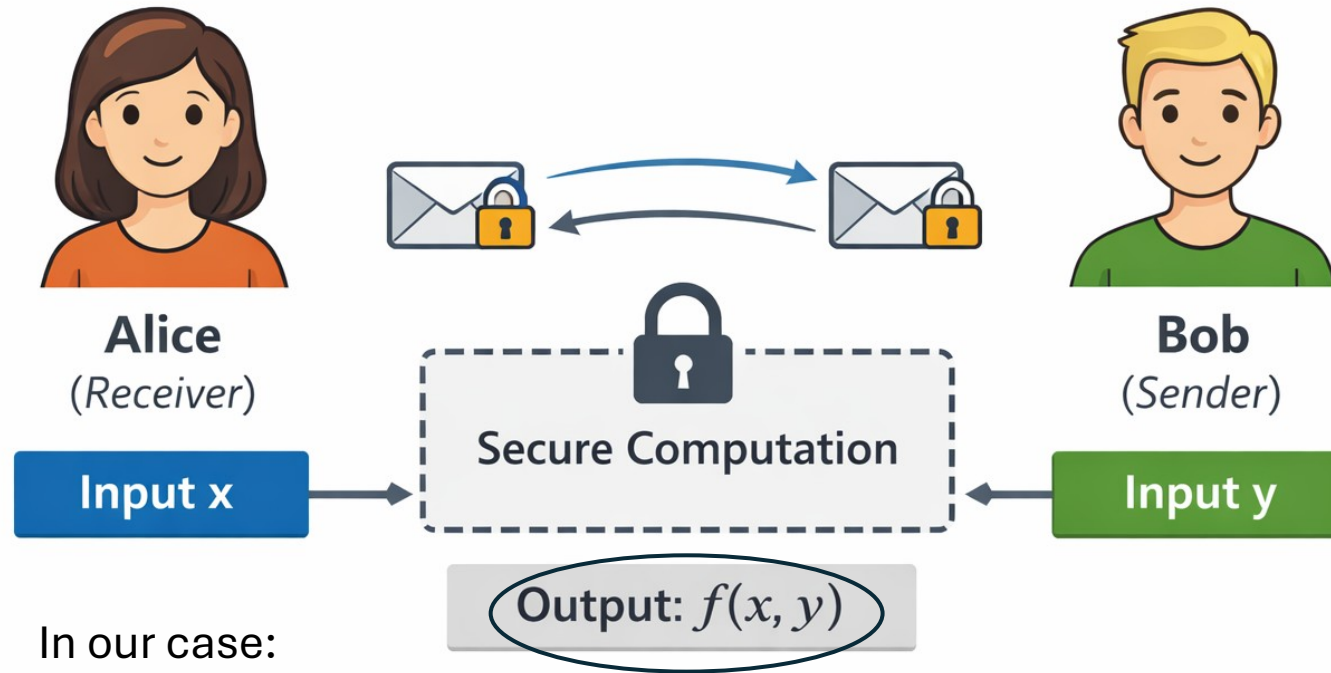
¹Technion and AWS*, ²University of Toronto

May 14, 2026

Secure Two-Party Computation

Alice learns nothing
beyond x and $f(x,y)$

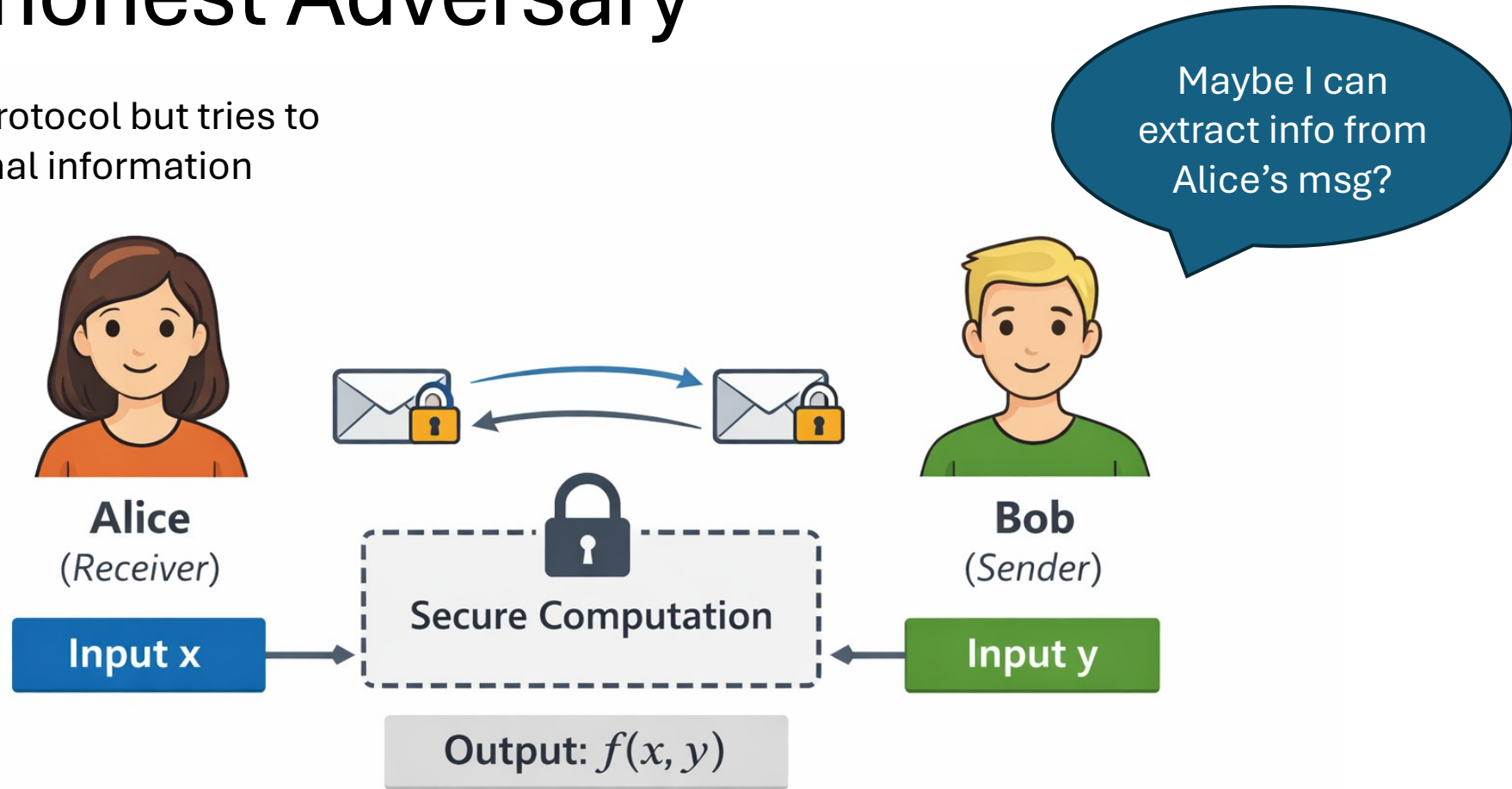
Bob learns nothing
beyond y



In our case:
only Alice
learns the
output.

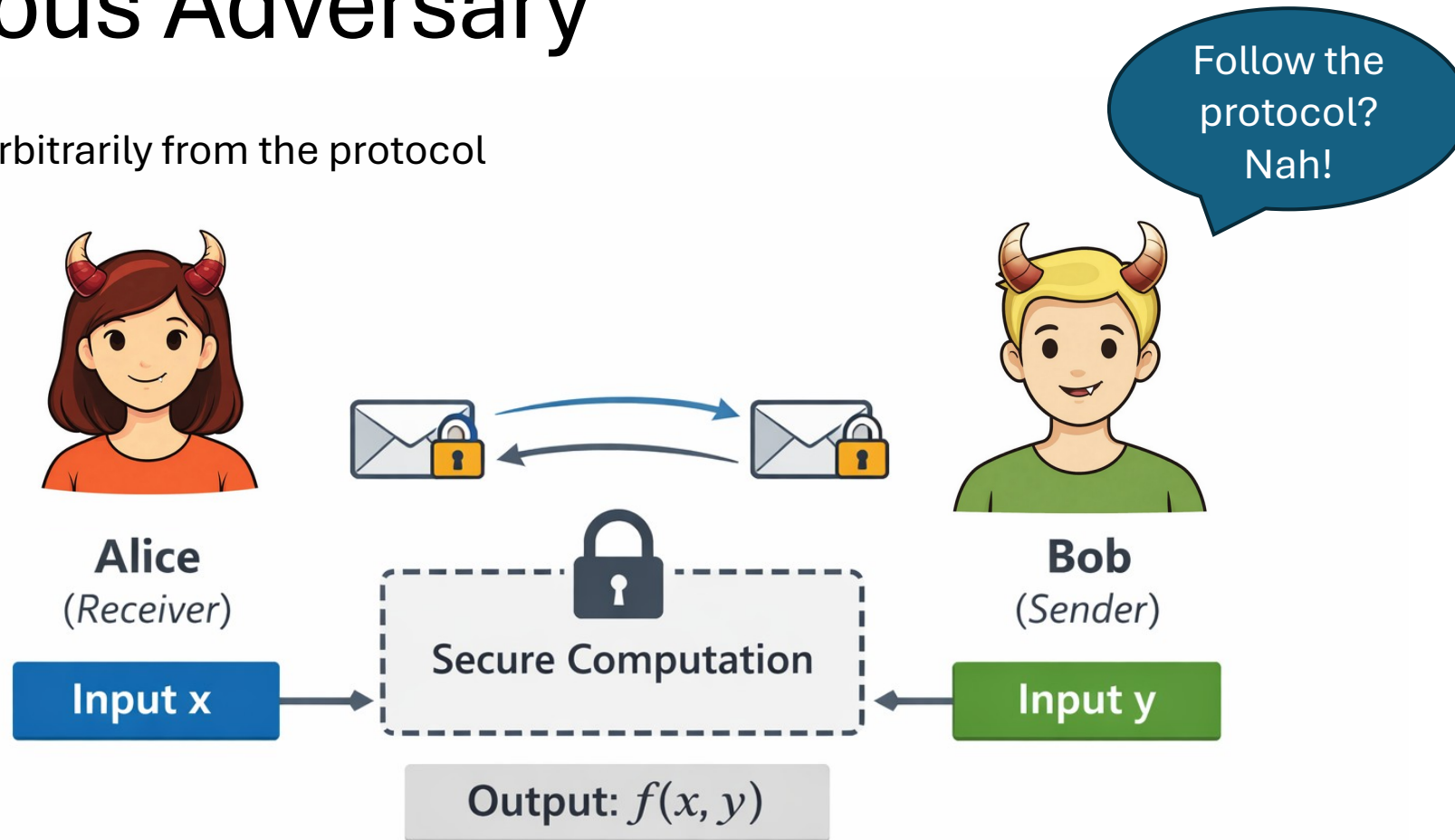
Semi-honest Adversary

Follows the protocol but tries to learn additional information from its view



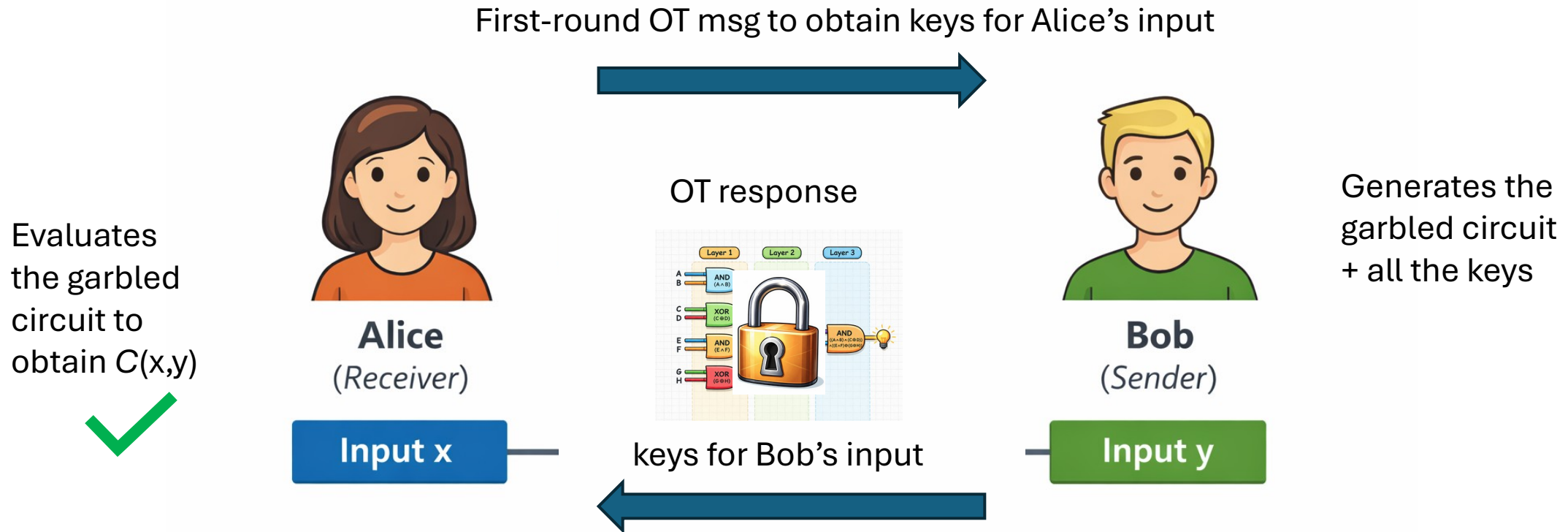
Malicious Adversary

Can deviate arbitrarily from the protocol



Garbled Circuits [Yao 86]

λ : computational security parameter



Semi-Honest

Two rounds (NISC)

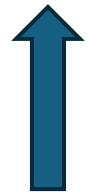
Communication: $O(|C|\lambda)$

Assume two-round OT

**Can malicious NISC protocols match
the communication complexity of the
semi-honest Yao's garbled circuits?**

Generic Transformation [GMW 87]

- Using a generic non-interactive zero-knowledge proof, we can upgrade semi-honest security to malicious security.
 - Can be constructed in the random oracle model.
 - However, it makes a **non-black-box** use of cryptographic primitives.



less efficient, less modular

Can we achieve malicious security with only black-box use of cryptographic primitives?

Black-Box Compilers [IPS 08, IKSS 22]

- Malicious security
- Two rounds
- Assumptions: Random Oracle and OT correlations
- Communication: $O(|C|) \cdot \text{poly}(\lambda)$



Consistency checks for Verifiable Secret Sharing
incur a bottleneck in communication cost

**Can we construct a two-round,
malicious NISC protocol with constant
communication overhead $O(|C|\lambda)$?**

Our Result

Main Theorem

Assuming a random oracle and random bit OT correlations, there is a malicious-secure, two-party NISC protocol for any Boolean circuit C with a communication cost of $O(|C|\lambda)$ bits.

- ✦ Black-box use of underlying cryptographic primitives
- ✦ Compatible with FreeXOR optimization

Comparison

Work	Assumptions	Communication Cost	Security
[IKO ⁺ 11]	OT correlations + PRG	$O(C \lambda) \cdot \text{polylog}(C , \sigma)$	Standard
[AMPR14]	2-round batch-committing OT	$O(C \cdot \lambda\sigma)$	Standard
[IKO ⁺ 11, HIV17]	OT correlations + RO	$O(C \lambda)$	Correlated abort
[DILO22]	programmable tensor OLE + RO	$O(C \lambda)$	Standard
Ours	OT correlations + RO	$O(C \lambda)$	Standard

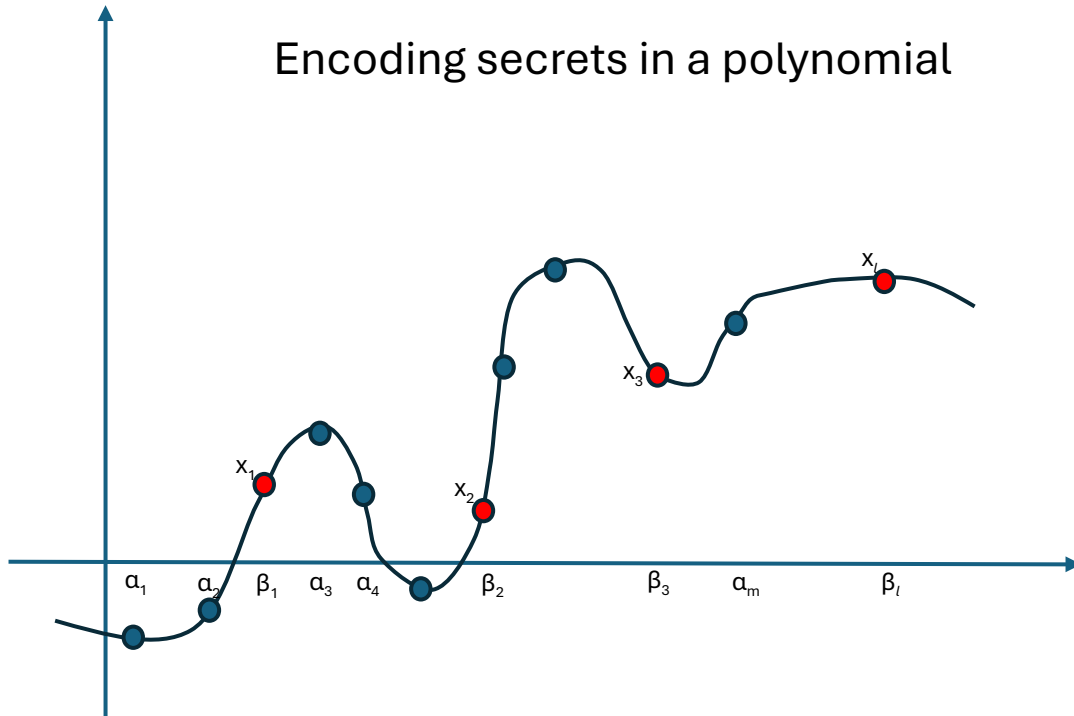
Table 1 : Comparison of NISC protocols in prior works. λ is an exact computational security parameter, and σ is a statistical security parameter.

Our Compiler:
A novel combination of
IPS Compiler + Packed SS + Ligerio Proofs

Packed Shamir Secret Sharing [FY 92]

Over finite field $\mathbb{F}$

Encoding secrets in a polynomial



l : packing size

t : corruption threshold

$\mathbf{x} = (x_1, x_2, \dots, x_l)$ secrets shared to m parties

Fix l distinct points: $\beta_1, \dots, \beta_l$ ← hide secrets

Fix m distinct points: $\alpha_1, \dots, \alpha_m$ ← generate shares

Share($\mathbf{x}$) algorithm:

1. Sample a random polynomial p of degree k conditioned on $p(\beta_1) = x_1, \dots, p(\beta_l) = x_l$
2. Distribute the shares by giving $p(\alpha_i)$ to party i

$[\mathbf{x}] := (p(\alpha_1), \dots, p(\alpha_m))$ is called a *sharing* of $\mathbf{x}$.

Multiplying two sharings gives
a sharing of (entry-wise) product of the secrets!

$$[\mathbf{x}]_{(t+l)} \cdot [\mathbf{y}]_{(t+l)} = [\mathbf{xy}]_{2(t+l)}$$

Degree-2 Polynomials

- We focus on degree-2 polynomials, as there are standard techniques [IK 00] in the two-party setting to extend this to general circuits.
- Alice has input vector $\mathbf{x} = (x_1, x_2, x_3)$.
- Bob has input vector $\mathbf{y} = (y_1, y_2, y_3, y_4)$.
- Goal: compute the degree-2 polynomial

$$p(\mathbf{x}, \mathbf{y}) = 1 + y_1 + x_1 y_3 + x_2^2 + x_3 y_2 + y_3 y_4$$

Decomposition

$$p(\mathbf{x}, \mathbf{y}) = 1 + y_1 + x_1 y_3 + x_2^2 + x_3 y_2 + y_3 y_4$$

Vector of monomials p :

1	y_1	$x_1 y_3$	x_2^2	$x_3 y_2$	$y_3 y_4$
---	-------	-----------	---------	-----------	-----------

Alice's component p^x :

1	1	x_1	x_2^2	x_3	1
---	---	-------	---------	-------	---

✖ ✖ ✖ ✖ ✖ ✖

Bob's component p^y :

1	y_1	y_3	1	y_2	$y_3 y_4$
---	-------	-------	---	-------	-----------

$$\mathbf{x} = (x_1, x_2, x_3)$$



Alice
(Receiver)



Assign values

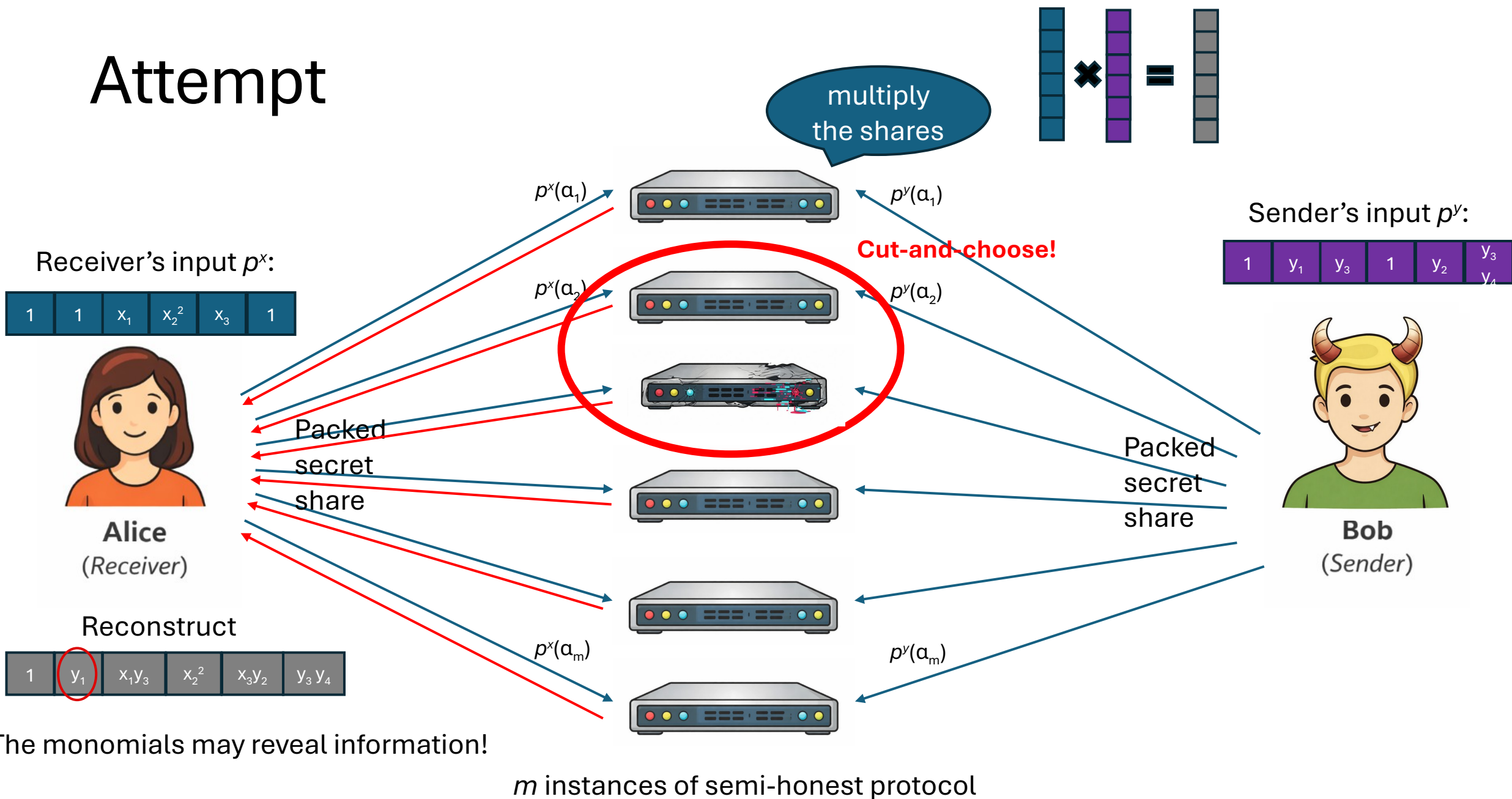


Bob
(Sender)



$$\mathbf{y} = (y_1, y_2, y_3, y_4)$$

Attempt



OT ← MFE Oblivious Linear Evaluation (OLE)

$$p^x(a_i) \cdot p^y(a_i) + p^z(a_i)$$

Sample a random vector p^z :

z_1	z_2	z_3	z_4	z_5	z_6
-------	-------	-------	-------	-------	-------

where $z_1 + \dots + z_6 = 0$.

1	1	x_1	x_2^2	x_3	1
---	---	-------	---------	-------	---



Alice

(Receiver)

Reconstruct

1	y_1	$x_1 y_3$	$x_2^2 y_2$	$x_3 y_4$	$y_3 y_4$
---	-------	-----------	-------------	-----------	-----------

masked by

z_1	z_2	z_3	z_4	z_5	z_6
-------	-------	-------	-------	-------	-------

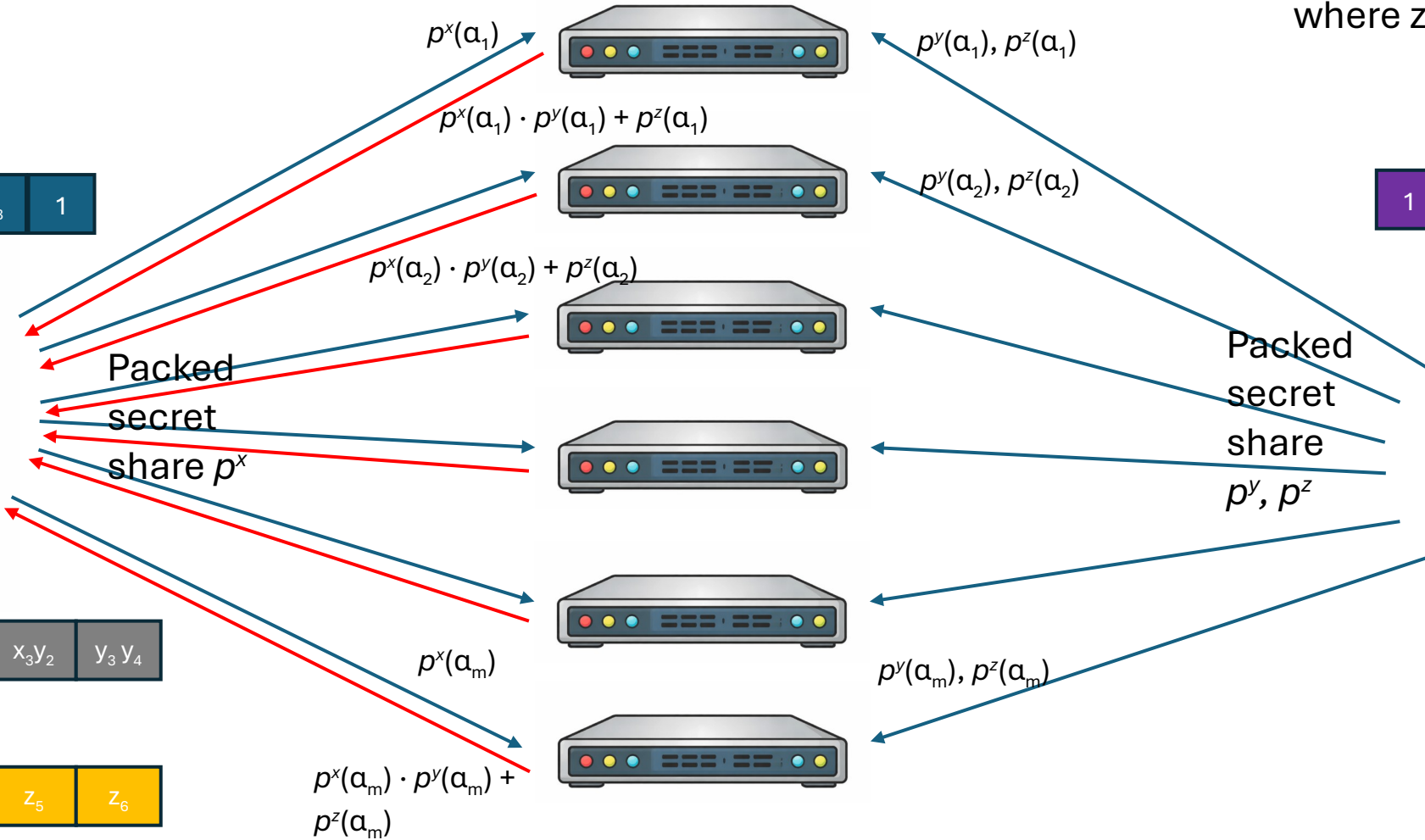
But the sum is still $p(x,y)$ ✓

1	y_1	y_3	1	y_2	y_4
---	-------	-------	---	-------	-------



Bob

(Sender)



Follow the
protocol?
Nah!



**However, a malicious adversary may
cheat in decomposition and
packed secret sharing.**

**Specialized zero-knowledge proofs to
enforce the behavior of malicious adversaries.**

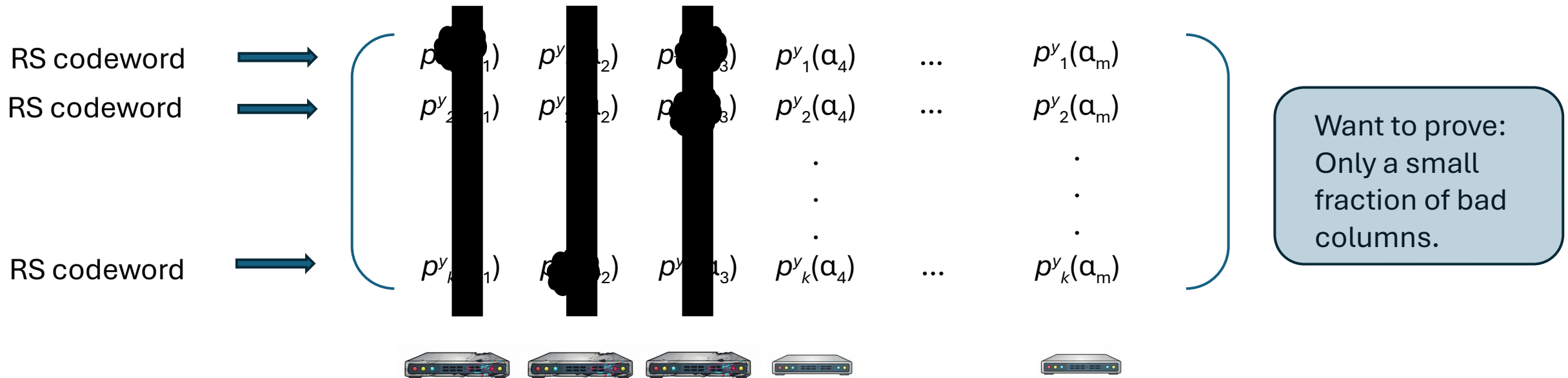
Correctness of Packed Secret Sharing

Shares are evaluations of $p(a_1), \dots, p(a_m)$ $\longleftrightarrow$ Reed-Solomon Code

Not too many corrupted shares $\longleftrightarrow$ Proximity of RS codewords

Suppose Bob shares $k = O(|C| / l)$ vectors $p^y_1, p^y_2, \dots, p^y_k$.

Let's put their packed Shamir sharings row-by-row into a matrix:



Even if one share is incorrect, the semi-honest instance will compute a wrong result

Ligero Proofs [AHIV 17]

$$|\mathbb{F}| = 2^\lambda$$

Random coefficients $\rightarrow$

$$\begin{matrix} r_1 \\ r_2 \\ \vdots \\ r_k \end{matrix} \begin{pmatrix} p^y_1(a_1) & p^y_1(a_2) & p^y_1(a_3) & \dots & p^y_1(a_m) \\ p^y_2(a_1) & p^y_2(a_2) & p^y_2(a_3) & \dots & p^y_2(a_m) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ p^y_k(a_1) & p^y_k(a_2) & p^y_k(a_3) & \dots & p^y_k(a_m) \end{pmatrix}$$

folding:

$$u_1 \quad u_2 \quad u_3 \quad \dots$$

$$\begin{pmatrix} p^y_1(a_m) \\ p^y_2(a_m) \\ \vdots \\ p^y_k(a_m) \end{pmatrix} \quad u_m$$

Non-Interactive:
Use RO to do
Fiat-Shamir

linear combination $\leftarrow$



Prover:

1. commits the matrix column-wise
2. computes $(u_1, \dots, u_m) = r_1 \cdot p^x_1 + r_2 \cdot p^x_2 + \dots + r_k \cdot p^x_k$
3. opens a random subset of columns



Verifier checks:

1. whether $(u_1, \dots, u_m)$ is a valid RS codeword
2. whether the opened columns follow the computation

accepts $\rightarrow$

Very few corrupted columns ✓

Proximity Gap Theorem [BCIKS 20]

Enforce Malicious Behavior



A malicious adversary may not perform the packed secret sharing honestly!

Ligero Proofs for Reed-Solomon Proximity!



A malicious adversary may not follow the decomposition rule!

Ligero Proofs for Linear Constraints!
(see paper)



The original Ligero paper has proofs for row-wise proximity, but our construction requires proofs for a column-corruption guarantee.

Summary

- **IPS Compiler:** runs m parallel semi-honest instances; cut-and-choose limits cheating to a small fraction of instances
- **Packed Secret Sharing:** amortizes the communication cost
- **Ligero Proofs:** enforce valid sharing and decomposition



Alice
(Receiver)



Two-round **semi-honest** protocol per instance



Two-round compiled **malicious** protocol



Bob
(Sender)



Communication Cost

1. We run m parallel semi-honest instances.
2. Cost per semi-honest instance is $O(|C|\lambda / l)$ and set $l = O(m)$.
3. Ligerio proofs + cut-and-choose: $O(|C|\lambda)$.
4. Total: $m \cdot O(|C|\lambda / l) + O(|C|\lambda) = O(|C|\lambda)$.

Matching semi-honest Yao!

Future Work

1. Extend the 2PC to multiple parties. Under similar assumptions, can we have a two-round malicious MPC with low communication cost?
2. If the communication channel is bi-directional, can we make both parties output in two rounds with the same communication complexity?
3. Can we make the protocol more concretely efficient for real-world applications?

Takeaway

**Assuming RO and OT correlations,
malicious NISC can match the communication
complexity of semi-honest Yao.**

Questions?