Concretely Efficient Large-Scale MPC with Active Security (or, TinyKeys for TinyOT)

Carmit Hazay, Emmanuela Orsini, Peter Scholl and **Eduardo Soria-Vazquez**

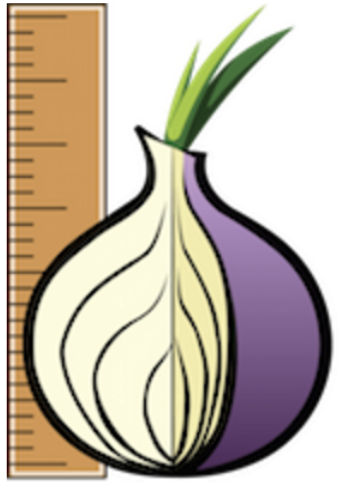
University of BRISTOL

ECRYPT

Large-Scale MPC

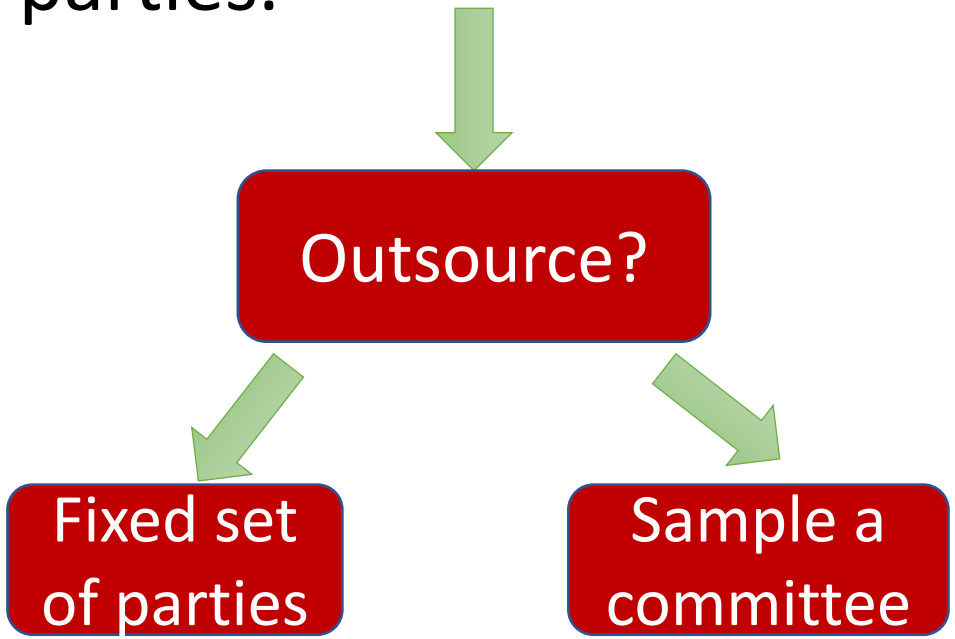
Growing number of users want to compute ***privately*** and ***jointly***.

1229 farmers (auction)



+6000 relays (statistics)

Current practical MPC **doesn't scale well** for large numbers of parties.



MPC setting in this talk

Main focus:

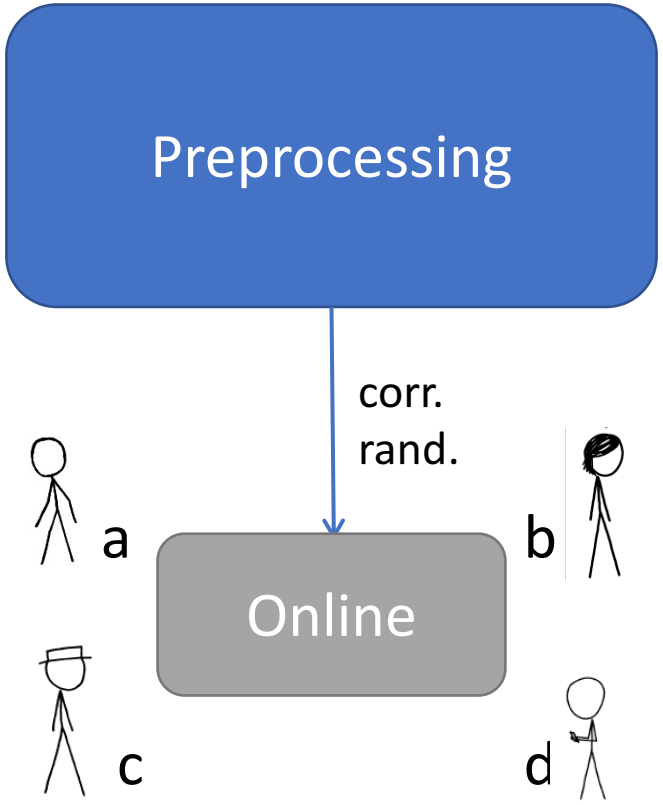
- **Concrete efficiency** for **large numbers** of parties (e.g. n in 10s, 100s).

Adversary:

- Static, **active**.
- **Dishonest majority**, but **not full threshold**!
  - Assume $h > 1$ honest parties to increase efficiency.

Model of Computation:

- Boolean circuits.
- Preprocessing phase.



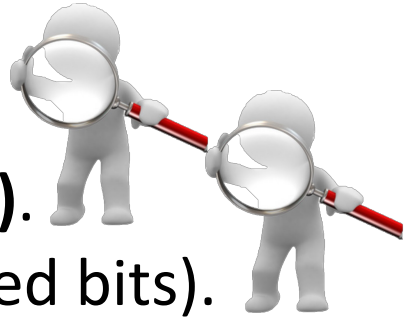
Our results

New TinyOT-style protocol (actively secure, dishonest majority) exploiting more honest parties:

- Up to **34x less communication** compared with [WRK17]'s TinyOT with $n-1$ corruptions.
- Up to **18x less communication** compared with [WRK17]'s TinyOT mixed with committees ($h > 1$ honest parties).
- Good improvements (2-6x less comm) with just **10% honest parties**.

How to scale TinyOT

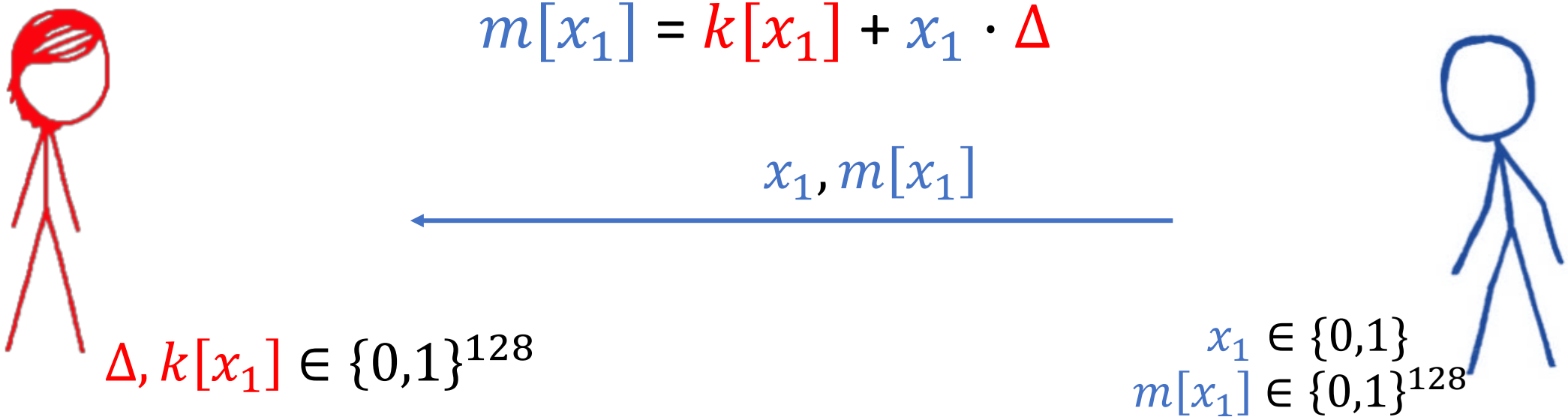
The TinyOT protocol [NNOB12]



- Based on additive secret sharing: $x = x_1 + x_2$.
- Multiplications computed using Beaver's triples: **(x, y, xy)**.
- Active security: Information-theoretic MACs (authenticated bits).

The TinyOT protocol [NNOB12]

- Based on additive secret sharing: $x = x_1 + x_2$.
- Multiplications computed using Beaver's triples: **(x, y, xy)**.
- Active security: Information-theoretic MACs (authenticated bits).



The TinyOT protocol [NNOB12]

- Based on additive secret sharing: $x = x_1 + x_2$.
- Multiplications computed using Beaver's triples: **(x, y, xy)**.
- Active security: Information-theoretic MACs (authenticated bits).

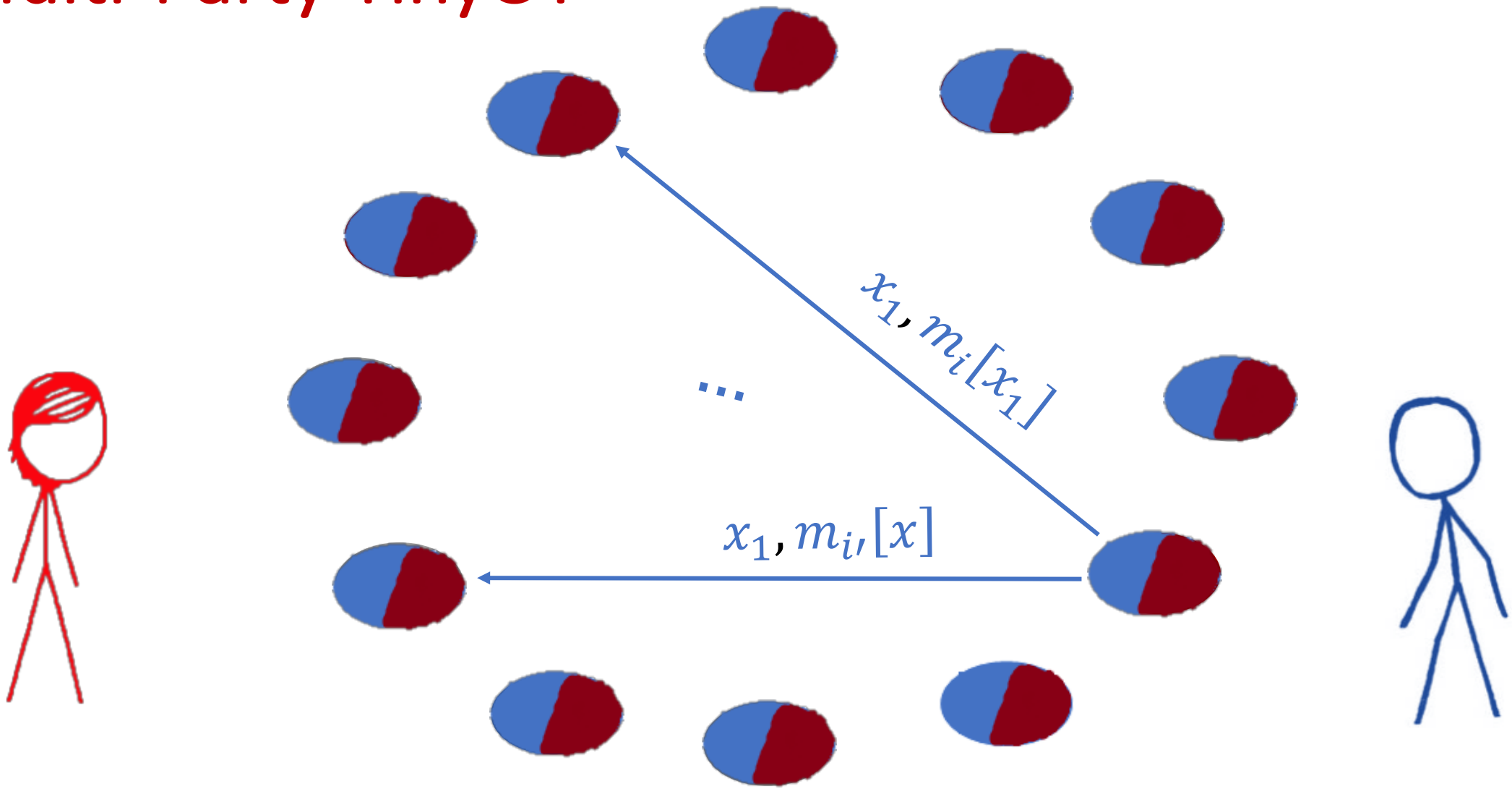
$$m[x_2] = k[x_2] + x_2 \cdot \Delta$$

$$x_2, m[x_2] \longrightarrow$$

$x_2 \in \{0,1\}$
$m[x_2] \in \{0,1\}^{128}$

$\Delta, k[x_2] \in \{0,1\}^{128}$

Multi-Party TinyOT

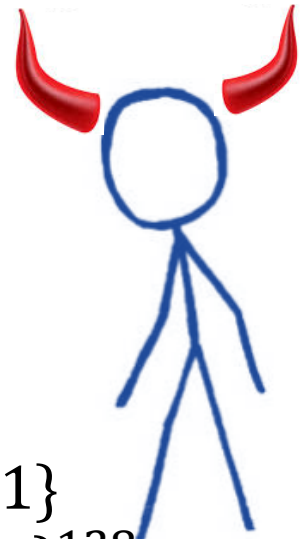


The TinyOT protocol [NNOB12]

- Based on additive secret sharing: $x = x_1 + x_2$.
- Multiplications computed using Beaver's triples: **(x, y, xy)**.
- Active security: Information-theoretic MACs (authenticated bits).

$$m[x_1] = k[x_1] + x_1 \cdot \Delta$$

$$x_1 + 1, m[x_1] + \Delta \longleftarrow$$



$\Delta, k[x_1] \in \{0,1\}^{128}$

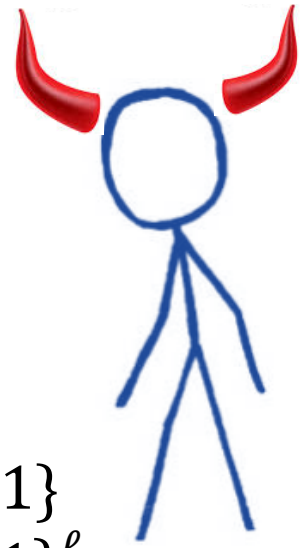
$x_1 \in \{0,1\}$
$m[x_1] \in \{0,1\}^{128}$

The TinyOT protocol [NNOB12]

- Based on additive secret sharing: $x = x_1 + x_2$.
- Multiplications computed using Beaver's triples: **(x, y, xy)**.
- Active security: Information-theoretic MACs (authenticated bits).

$$m[x_1] = k[x_1] + x_1 \cdot \Delta$$

$$x_1 + 1, m[x_1] + \Delta \longleftarrow$$



$\Delta, k[x_1] \in \{0,1\}^\ell$

$\ell \ll 128$

$x_1 \in \{0,1\}$
$m[x_1] \in \{0,1\}^\ell$

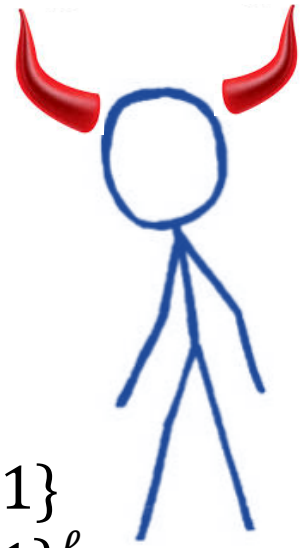
The TinyOT protocol [NNOB12]

- Based on additive secret sharing: $x = x_1 + x_2$.
- Multiplications computed using Beaver's triples: **(x, y, xy)**.
- Active security: Information-theoretic MACs (authenticated bits).

$\overline{m}[x_1]$

Divided costs by $128/\ell$

$x_1] + \Delta$



$\Delta, k[x_1] \in \{0,1\}^\ell$

$\Delta, k[x_1] \in \{0,1\}^\ell$

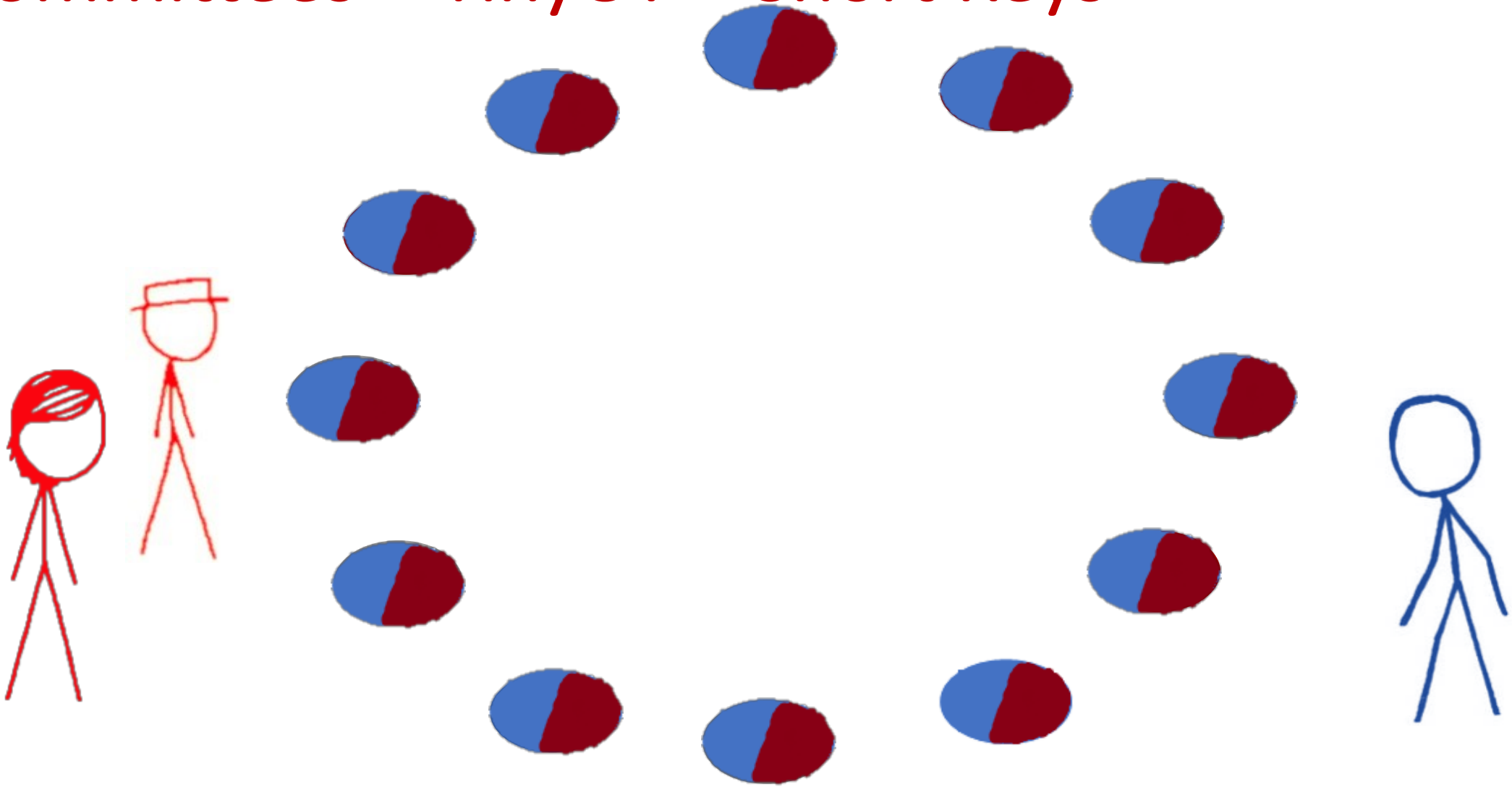
$\ell \ll 128$

$\ell \cdot h \geq s$

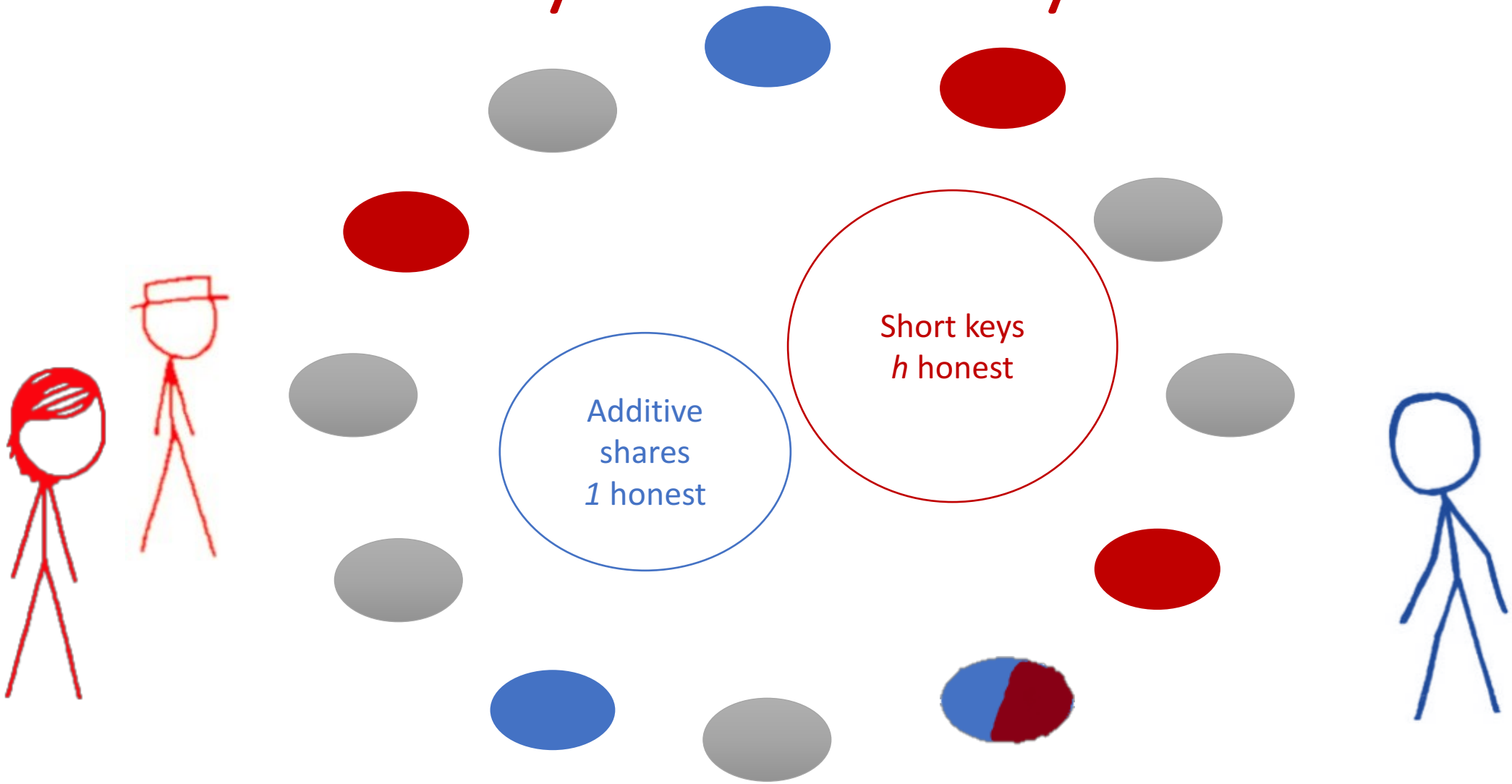
$x_1 \in \{0,1\}$

$m[x_1] \in \{0,1\}^\ell$

Committees + TinyOT + Short Keys



Committees + TinyOT + Short Keys



The problem with short MACs

$\mathbf{y} \in \{0,1\}^r$

$x_1 y_1 + s_1, \dots, x_r y_r + s_r$

$r\times$ Triple
$(\mathbf{x}, \mathbf{y}, \mathbf{xy})$

$\mathbf{x} \in \{0,1\}^r$

$s_1, \dots, s_r$

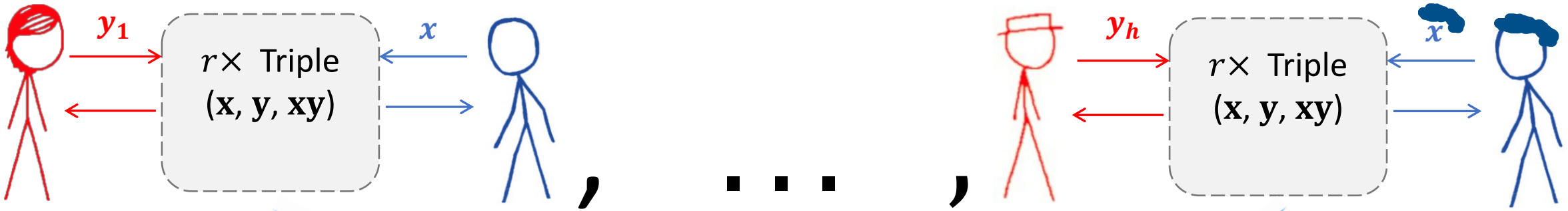
$L(\boldsymbol{y}) \approx H(\Delta) + \boldsymbol{y}$

$y_1, \dots, y_r \in \{0,1\}$
$\Delta \in \{0,1\}^\ell$
$k[x_1], \dots, k[x_r] \in \{0,1\}^\ell$

$x_1, \dots, x_r \in \{0,1\}$
$m[x_1], \dots, m[x_r] \in \{0,1\}^\ell$
$s_1, \dots, s_r \in \mathcal{U}(\{0,1\})$

Only 2^ℓ possible values for Δ !
ℓ as small as 1 !

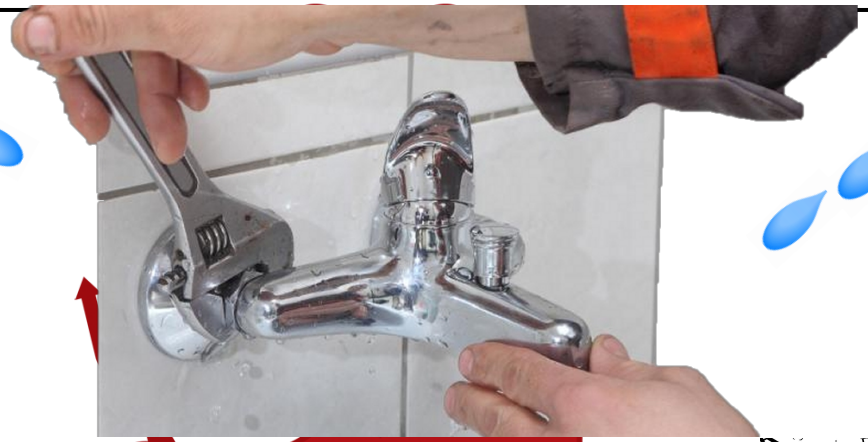
Leakage gets worse…



$L(y_1) \approx H(\Delta_1) + y_1$

$L(y_h) \approx H(\Delta_h) + y_h$

Use TinyKeys! [HOSS18]



$$L(y_1 + \cdots + y_h) = \sum_{i=1..h} H(\Delta_i) + y_i \approx$$

What is TinyKeys? [HOSS18]

- New tool for **large-scale MPC** (more honesty ⇒ shorter keys).
  - Base security on the concatenation of honest parties' keys.
- Security reduces to **Regular Syndrome Decoding**:
  - Not much easier than Syndrome Decoding ⇔ LPN.

[BM17] [Kir11]
[Saa07] [FS09] [BLN+09]
[BLP08] [BJMM12] [MO15]
[MS09] [CJ04] [MMT11] [NCB11]
[BLP11]

- Params: # products r, key length ℓ, # honest parties h.
- **Statistically hard** for small r/large h.

Protocol in Tiny Keys? [KOS18]

- New tool for **large-scale MPC** (more honesty ⇒ shorter keys).
  - Base security on the concatenation of honest parties' keys.

- Security reduces to **Regular Syndrome Decoding**:
  - Not much easier than syndrome decoding ⇔ LPN.

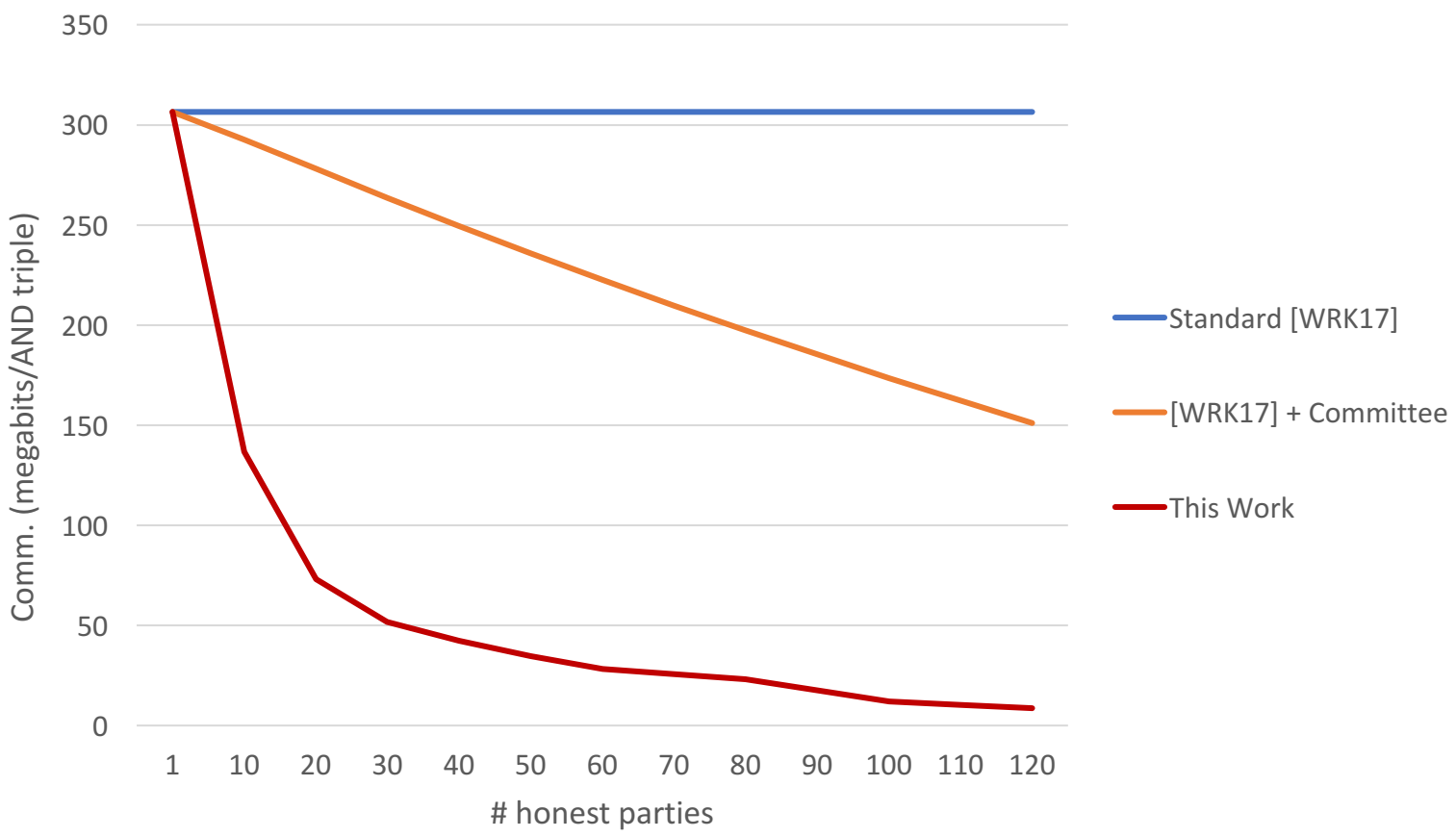
- Params: # products r, key length ℓ, # honest parties h.
- **Statistically hard** for small r/large h.

Problems with TinyKeys [HOSS18]

- Params: **# products r**, key length ℓ, # honest parties h.
  - A single Δ can only be used to produce *r* triples!
  - Solution: Use different ones for every *r* triples: $\Delta_{[0,r)}$, $\Delta_{[r,2r)}$, …
    Secure method for switching: $\Delta_{[0,r)} \rightarrow \Delta$, $\Delta_{[r,2r)} \rightarrow \Delta$, …

- Best bucketing technique cannot apply (mult. overhead: B).
  - Solution: Use previous bucketing techniques (mult. overhead: B^2).
  - Still worth! $B \in \{3,4\}$ in practice.

Communication complexity (400 parties)



Conclusion and future directions

- First extension of TinyKeys [HOSS18] to the **active setting**.
  - Take-away: Large-scale requires different/new techniques (bucketing, MACs).
- Improved TinyOT with 30+ parties.
  - Up to **18x** in communication (vs multiparty [WRK17] + committees).
  - Significant improvements (**2-6x**) with as little as **10% honest parties.**

Future challenges:

- Optimize TinyKeys: More **cryptanalysis** (conservative parameters atm).
- Adaptive adversaries? Actively secure TinyKeys-BMR [HOSS18]?

Thank you! Questions?

Paper: https://ia.cr/2018/843 [Full version]

Concretely Efficient Large-Scale MPC with Active Security (or, TinyKeys for TinyOT)

Carmit Hazay, Emmanuela Orsini, Peter Scholl and ***Eduardo Soria-Vazquez***

Mail: eduardo.soria-vazquez@bristol.ac.uk