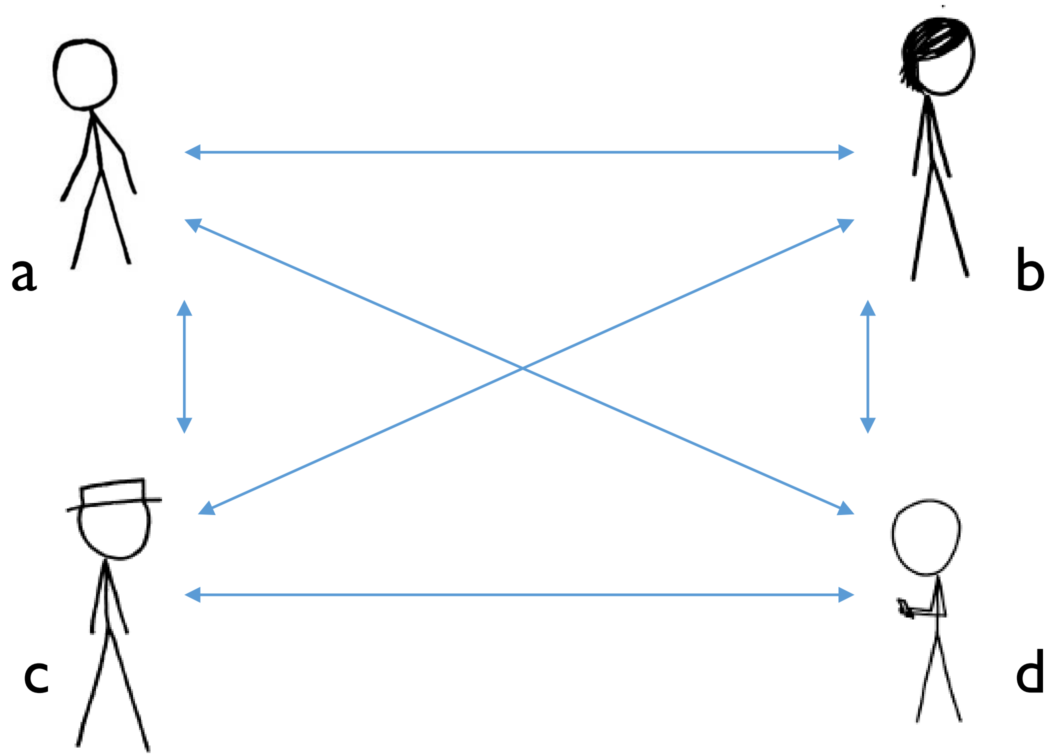
Efficient MPC From Syndrome Decoding

Or: Honey, I Shrunk the Keys

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

AARHUS UNIVERSITY

Secure Multi-Party Computation



Goal: Compute f(a,b,c,d)

Properties of secure computation protocols

- Computational model: Boolean/arithmetic circuits, RAM

- Adversary model:
  - **Passive** (semi-honest) or **active** (malicious)
  - **Threshold** t (number of corrupted parties)

- Efficiency:
  - Round/communication complexity
  - Computation

MPC setting in this talk

Main focus:

- Concrete efficiency for large numbers of parties

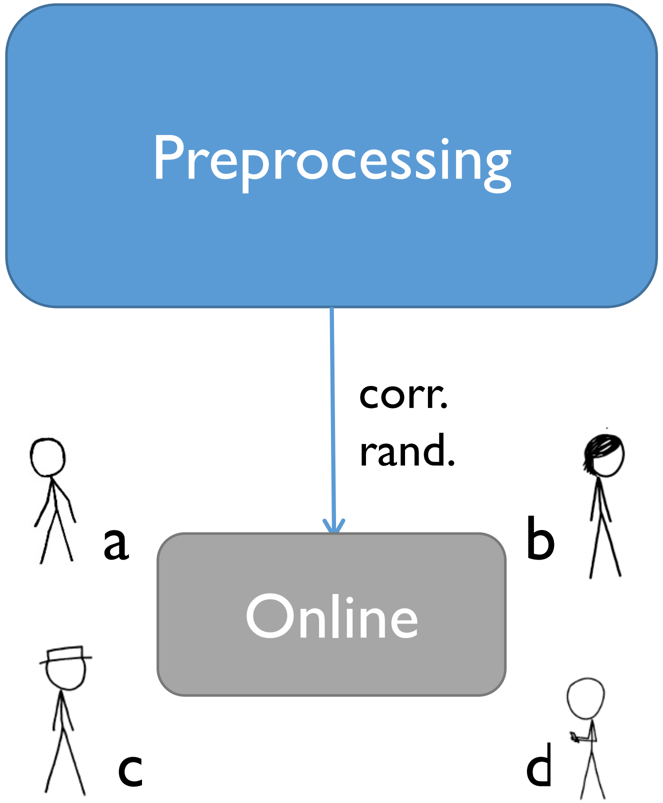
(e.g. n in 10s, 100s)

Adversary:

- Static, passive
- Dishonest majority ($t > n/2$)

Model of Computation:

- Boolean circuits
- Preprocessing phase



Motivation for large-scale, dishonest majority MPC

Large number of clients/users want to aggregate data, statistical analysis, surveys etc.

- E.g. statistics on Tor network activity, blockchain miners, app users etc.

MPC between all clients

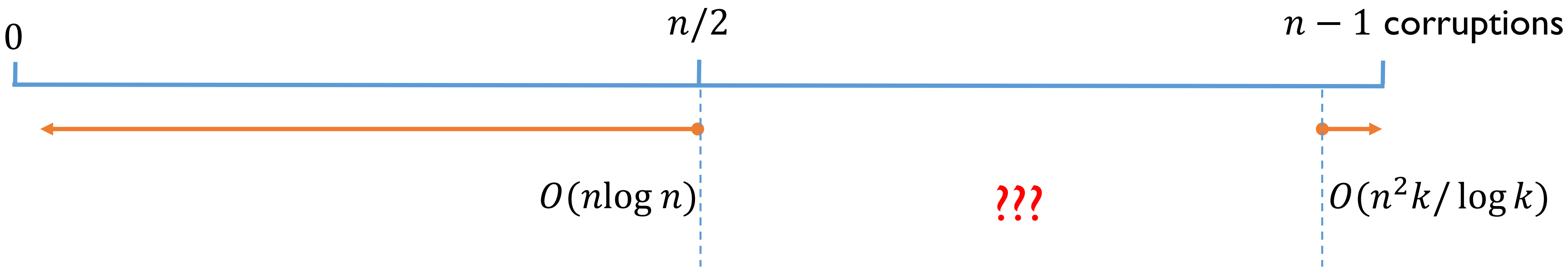
- Outsource to set of servers
- More servers = less trust

Main question

Can we trade off the number of corrupt parties for a more efficient, practical protocol?

AARHUS UNIVERSITY

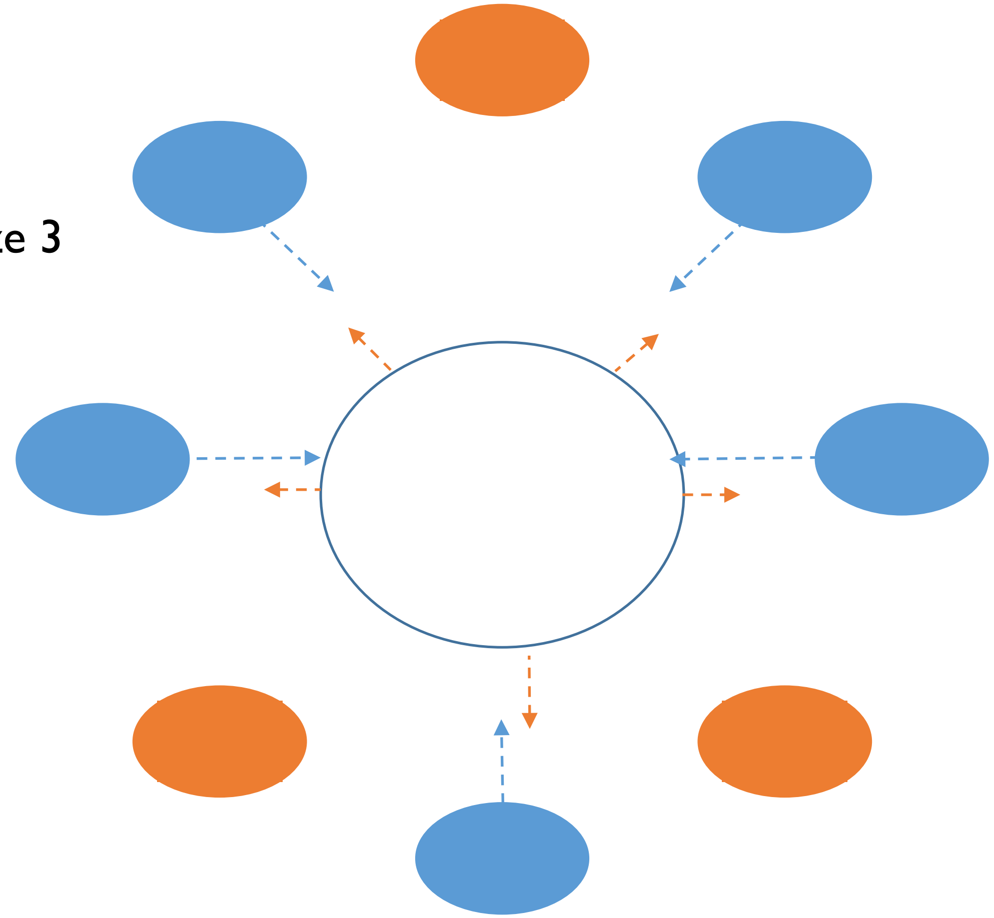
Corruption thresholds vs communication complexity of *practical* MPC



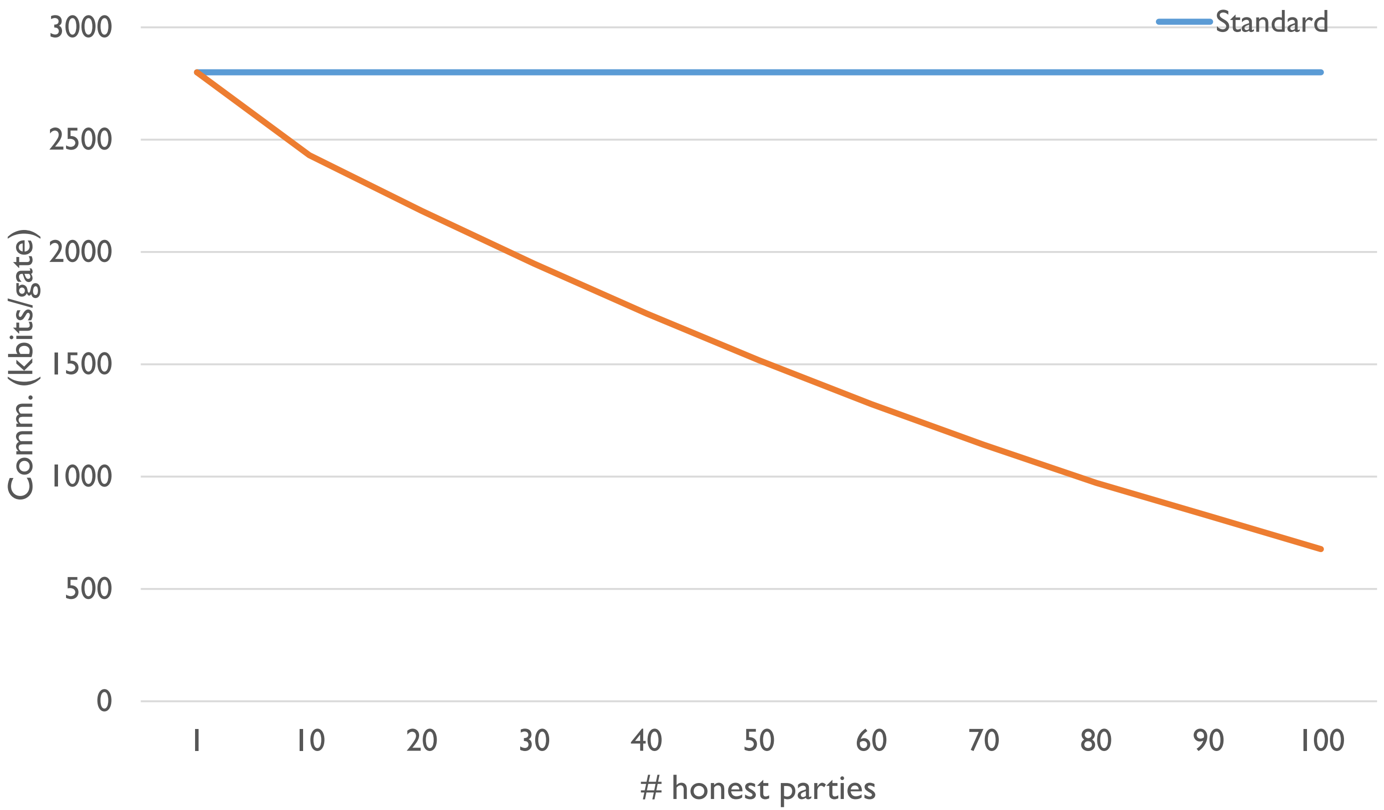
n parties, security k
Passive corruptions
Boolean circuits

Naive committee-based approach for t corruption

2 corruptions:
- Choose committee of size 3



Savings from naive committee approach with 200 parties and GMW protocol



variant of GMW by [Dessouky Koushanfar Sadeghi Schneider Zeitouni Zohner 17]

An asymptotically better approach using random committees

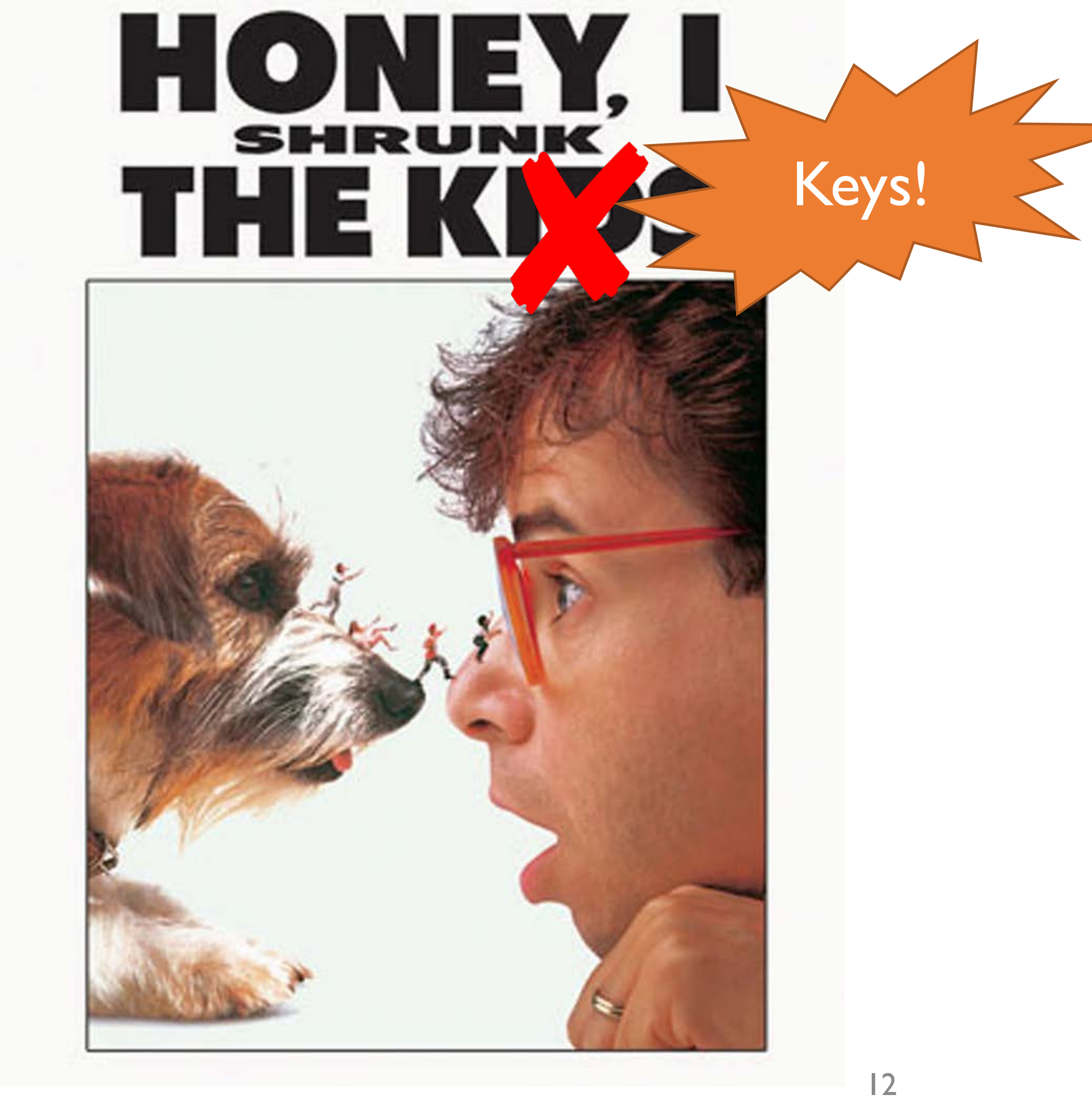
[Bracha '87]

- Suppose $t = \epsilon n$ for constant $\epsilon \in (0,1)$
- Sample random committee C of size k
  - C runs threshold-$(n-1)$ MPC protocol
  - Complexity: $O(k^3)$ per AND gate
- $\Pr[C$ is all corrupt$]$ is $\binom{t}{k} / \binom{n}{k}$
  - $\mathrm{negl}(k)$ for large enough n
  - k can be independent of n

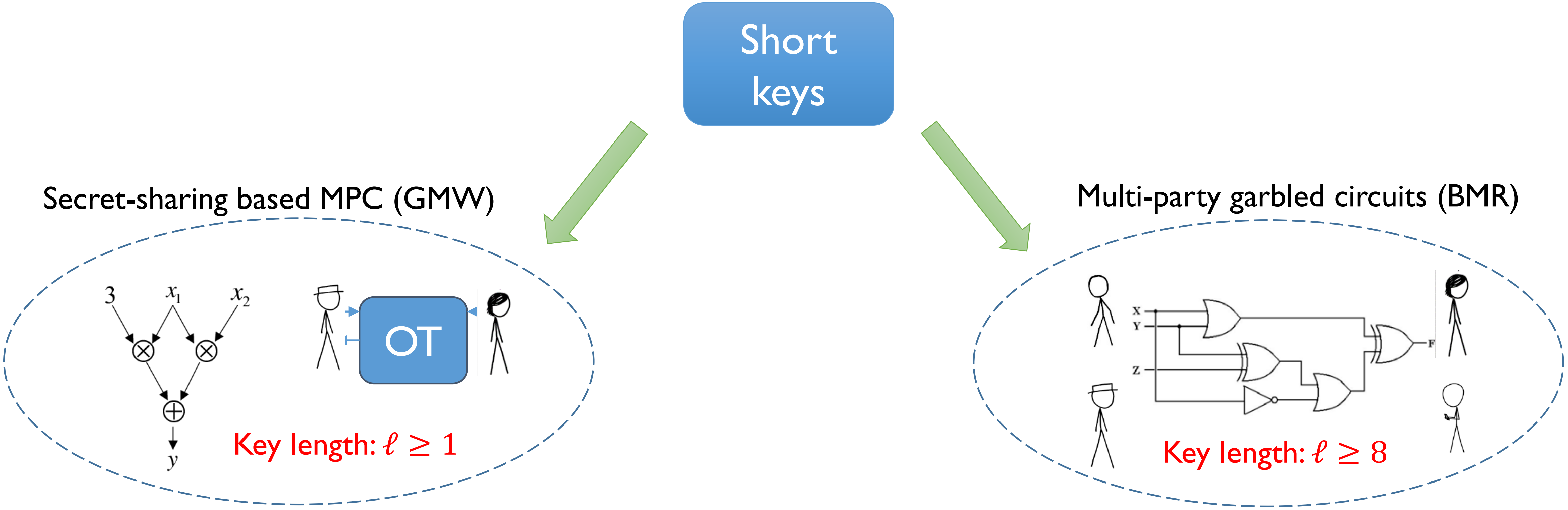
Can we do better? What about for smaller n?

New approach: short keys for secure computation

- Key idea:
  - "Weaken" existing protocol for $n-1$ corruptions by shrinking secret keys
  - Rely on concatenation of all honest parties' keys for security



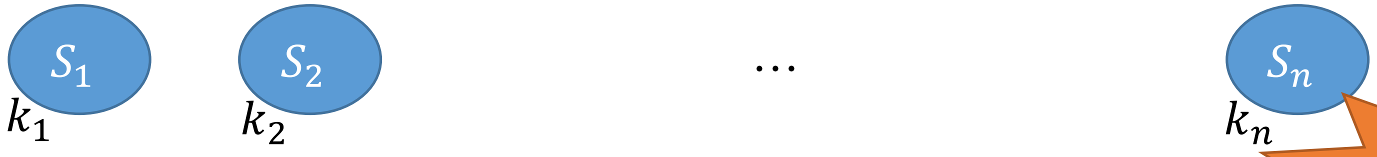
New MPC protocols with short keys and fewer corruptions



More honesty ⇒ shorter keys ⇒ more efficiency

Toy example: simple distributed encryption scheme

- Key distributed across n servers



Shrink the keys!

$$\text{Enc}(k_1, \ldots, k_n, x) = \sum_i H_i(k_i) \oplus m$$

- Hard to guess m if at least one $k_i \in \{0,1\}^\lambda$ is unknown

- What is h keys are unknown?
  - Can k_i be smaller?

AARHUS UNIVERSITY

Why should this work?

- Let $H_i: \{0,1\}^\ell \to \{0,1\}^r$ be a hash function
- Want

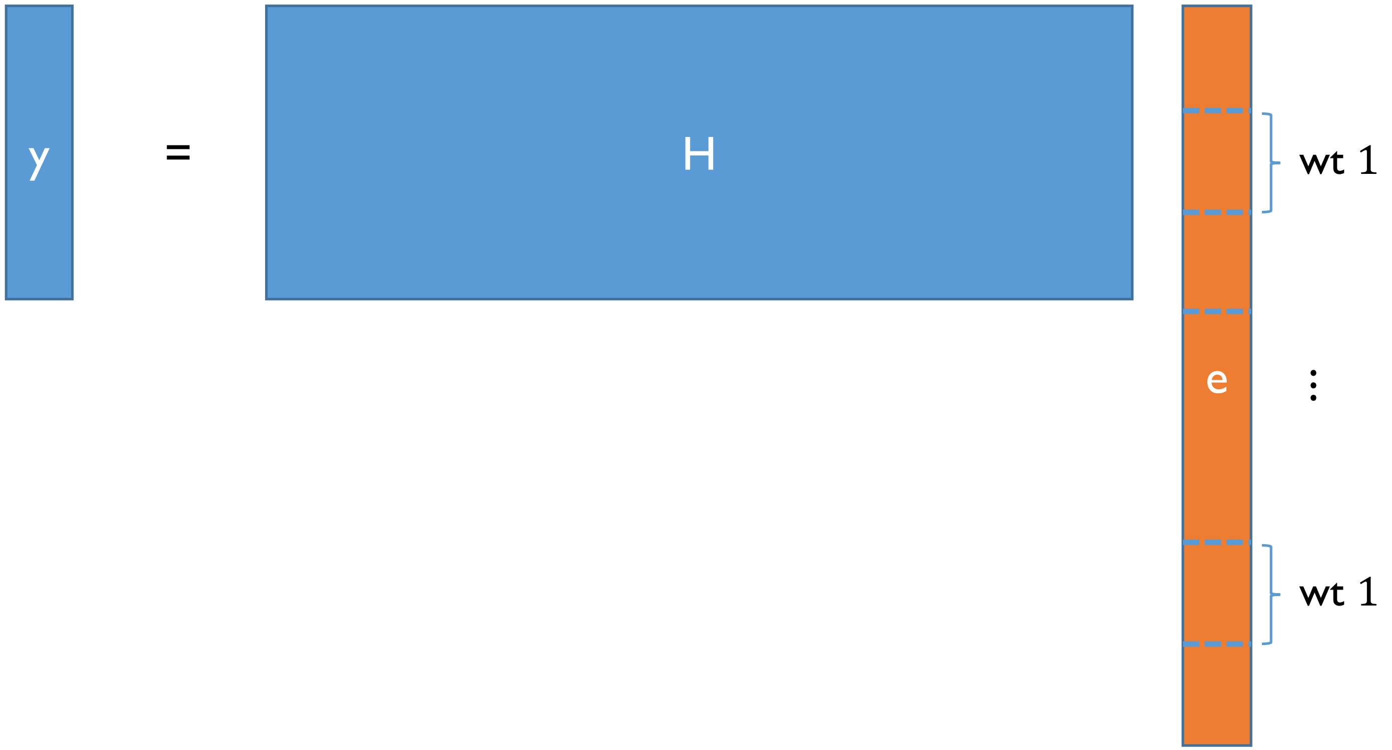
$$\sum_{i=1}^{n} H_i(k_i)$$

to be pseudorandom when $k_i \leftarrow \{0,1\}^\ell$ and h keys are unknown

AARHUS UNIVERSITY

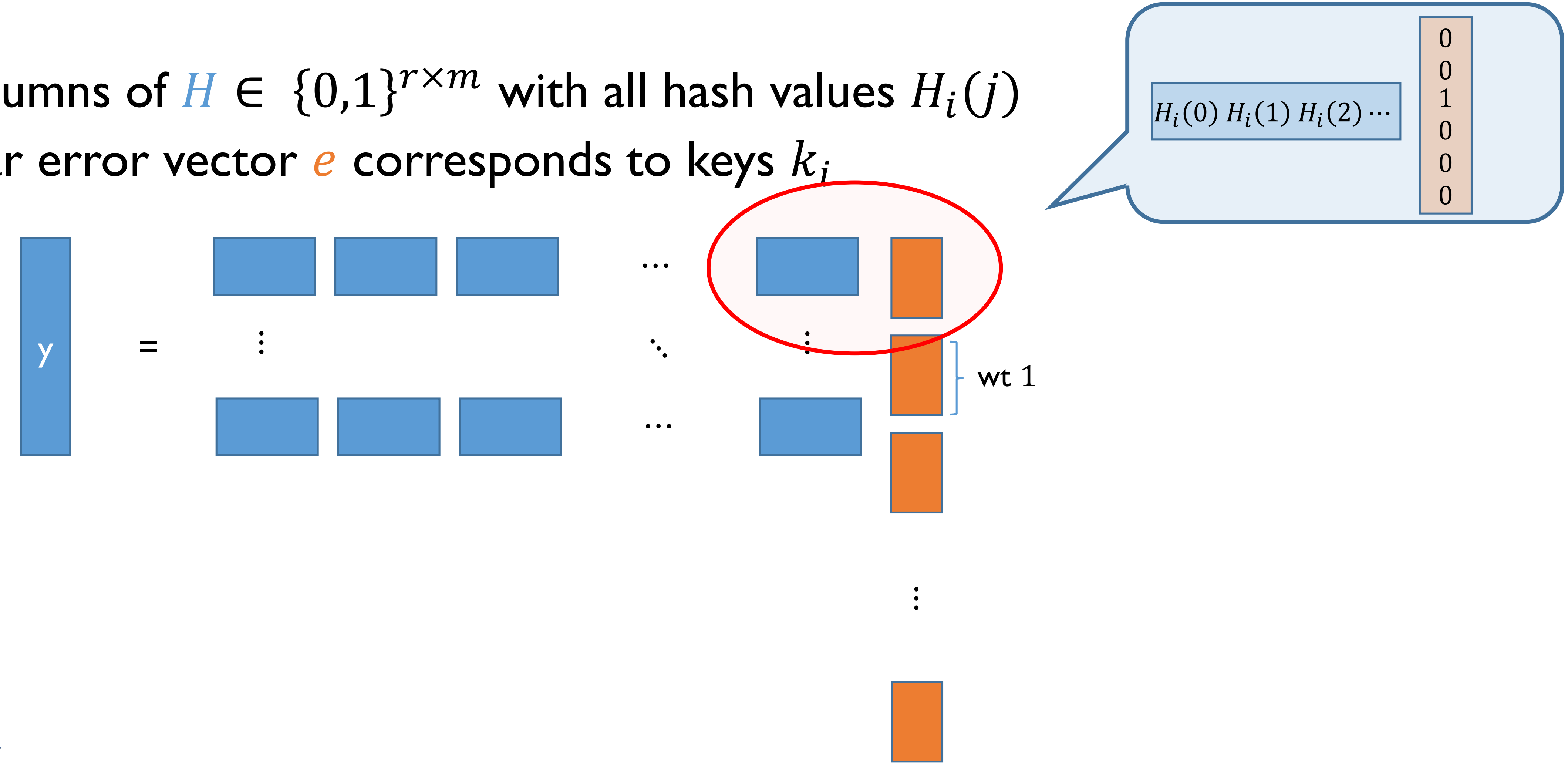
Regular syndrome decoding problem

- Sample random $H \in \{0,1\}^{r \times m}$, and regular $e \in \{0,1\}^m$ of weight h
- Given H and $y = He$, find e.



Equivalence of sum of hashes and regular syndrome decoding

- Fill columns of $H \in \{0,1\}^{r \times m}$ with all hash values $H_i(j)$
- Regular error vector e corresponds to keys k_i



Hardness of regular syndrome decoding

- Parameters:
  - Key length ℓ, # keys h, output length r

- Used for SHA-3 candidate FSB [Augot Finiasz Sendrier 03]
  - Not much easier than syndrome decoding ⇔ LPN

- Search-to-decision reduction
  (finding e as hard as distinguishing He from random)

- Statistically hard for small r/large h

Protocol 1: GMW-style MPC based on OT extension with short keys

[**G**oldreich **M**icali **W**igderson '87]

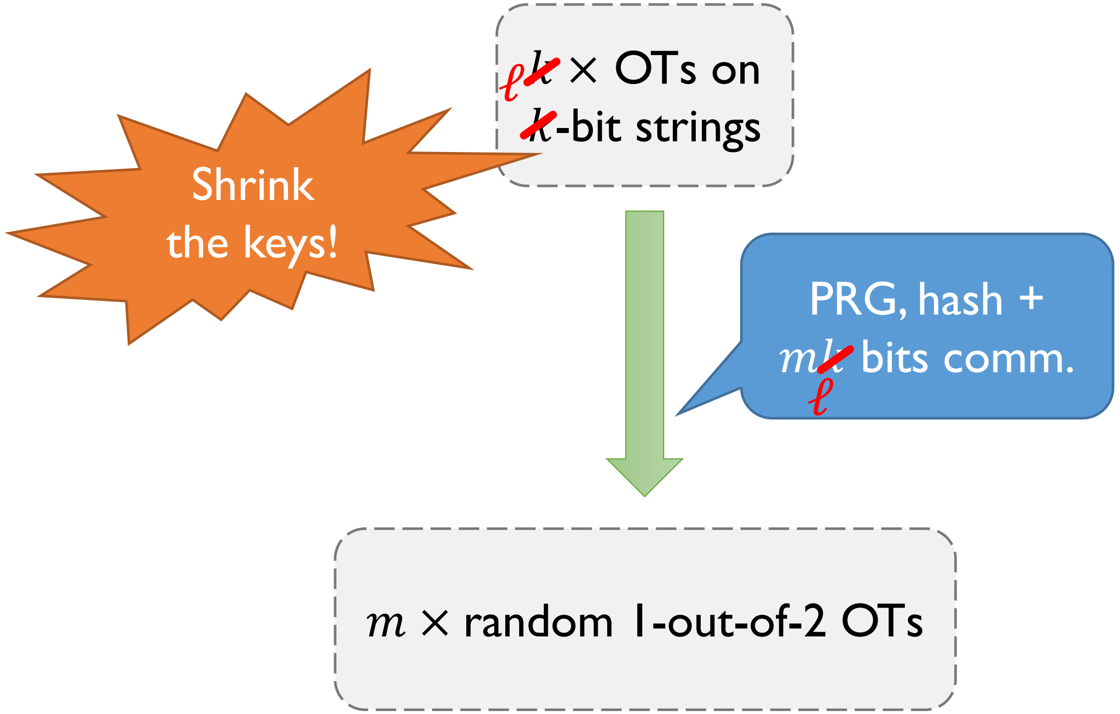
1-out-of-2 Oblivious Transfer

1-out-of-2 Oblivious Transfer gives secret-shared multiplication

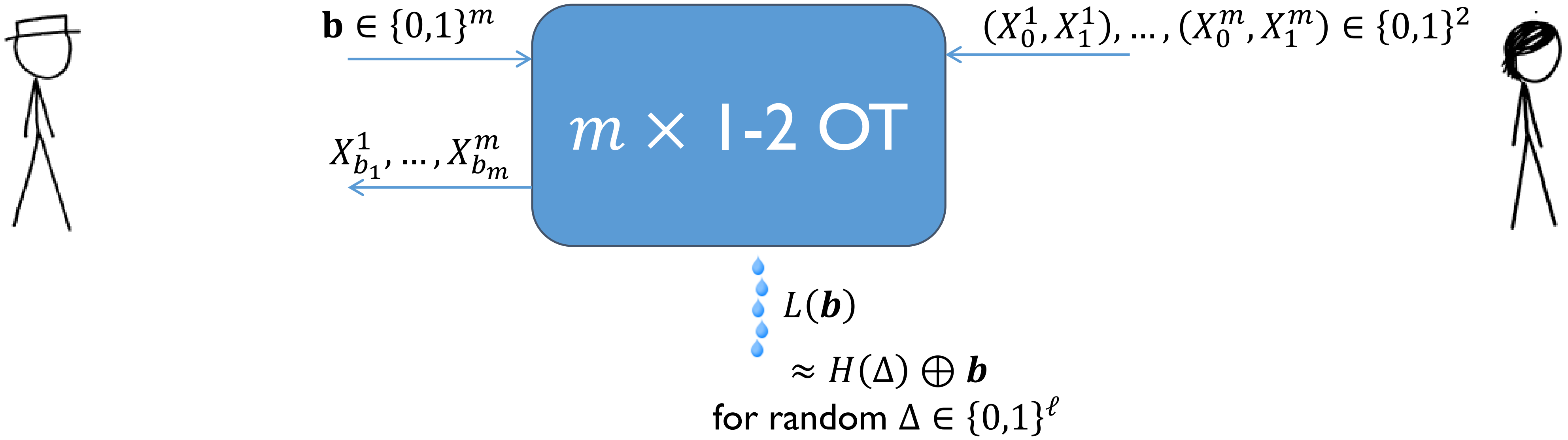
$$= (1 - b) \cdot X_0 + b \cdot X_1$$
$$= r + ab$$
$$= \underbrace{X_0}_{r} + b \cdot \underbrace{(X_1 - X_0)}_{a}$$

"IKNP" OT extension technique: converting k "seed" OTs into $m \gg k$ OTs

[**I**shai **K**ilian **N**issim **P**etrank 03]

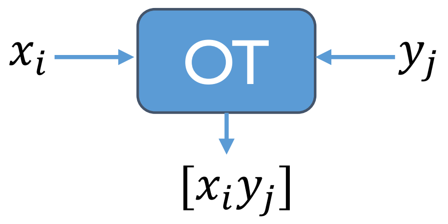


OT extension with short keys and leakage



Using leaky OT for GMW-style MPC

- First attempt: see what happens
  - Multiply shared $[x]$ and $[y]$ with GMW
  - Every pair (P_i, P_j):



  - Compute $[xy]$ from

$$xy = (x_1 + \cdots x_n)(y_1 + \cdots + y_n) = x_1 y_1 + \cdots x_i y_j + \cdots + x_n y_n$$

Problem: leakage on x_i with every corrupt party P_j

$\Rightarrow$ whp x_i leaks entirely if enough corruptions

Using leaky OT for GMW-style MPC

- Second attempt: rerandomize shares before multiplying
  - P_i inputs $(x_i + s_{ij})$ instead of x_i
  for random $s_{ij} \in \{0,1\}$
  such that $\sum_i s_{ij} = 0$

$$(x_1 + s_{11})y_1 + \cdots + (x_i + s_{ij})y_j + \cdots + (x_n + s_{nn})$$

$$= xy$$
$$+ (s_{11} + \cdots + s_{n1})y_1$$
$$\cdots$$
$$+ (s_{1n} + \cdots + s_{nn})y_n \quad = xy$$

What about the leakage?

- All inputs with leakage masked by shares of zero
- Only need to consider sum of all leakage on secret $x = \sum_i x_i$
- Leakage is equivalent to:

$$\sum_i H(i, \Delta_i) + x$$

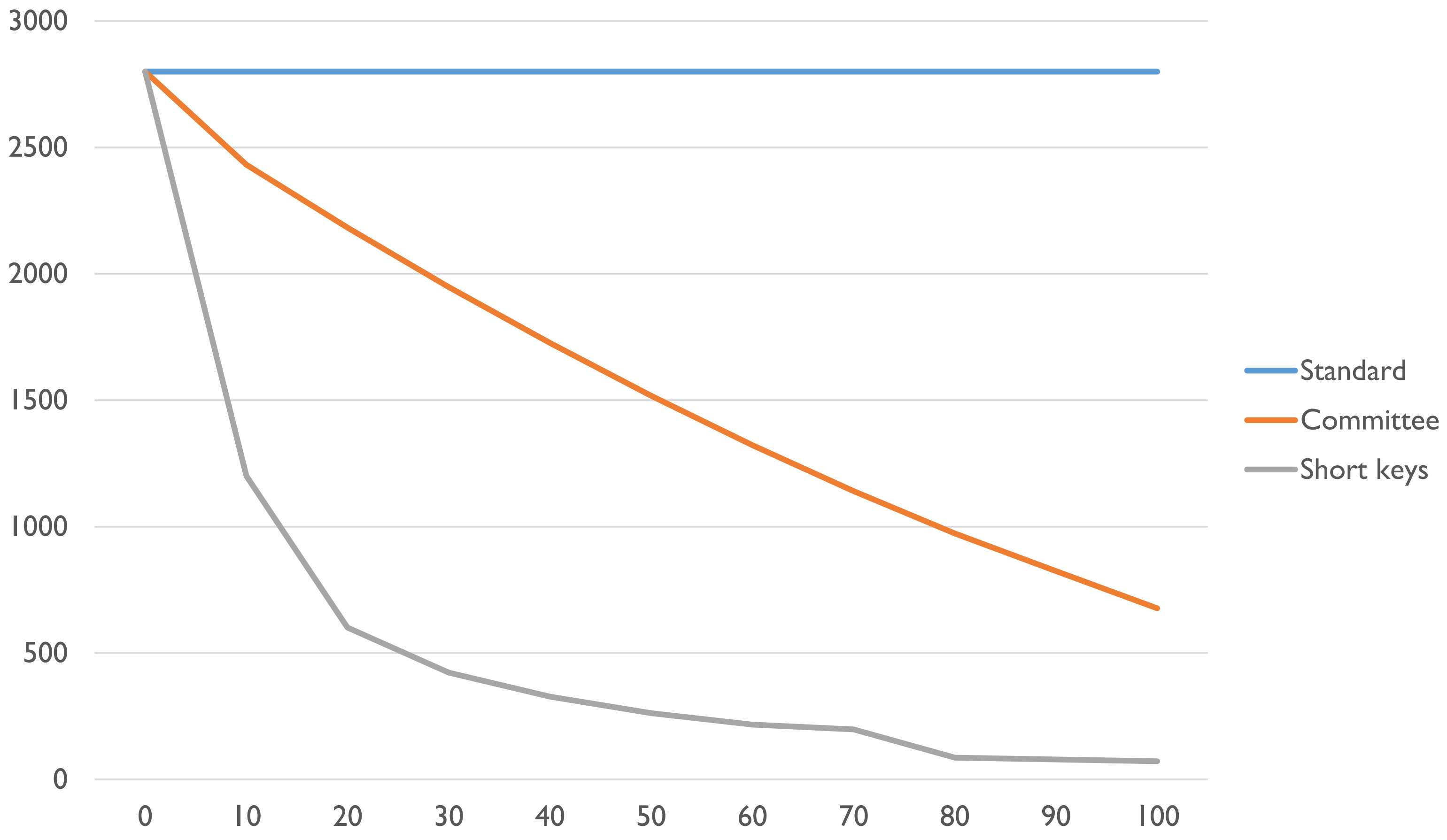
Pseudorandom by regular syndrome decoding assumption

Parameters and efficiency of GMW-based protocol

- Typically, each key can be used for $r = 300\text{-}500$ triples
- 1-bit keys when $h > s + r$ (e.g. $s = 40$ for stat. security)
  - Triple cost $\approx 3nt$ bits comm.
  - Assumes OT + OWF only (no RSD)

vs $O(n^2 k / \log k)$ for full-threshold

Reduction in communication from GMW with short keys (200 parties)

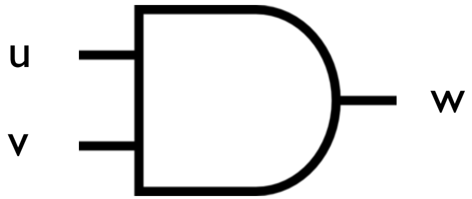


Protocol 2: BMR-based MPC based on multi-party garbled circuits with short keys

[**B**eaver **M**icali **R**ogaway ’90]

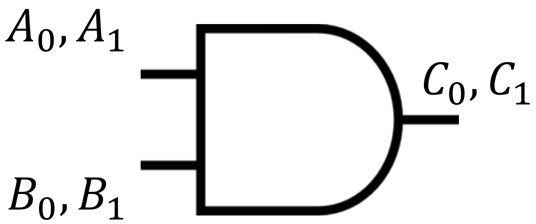
AARHUS UNIVERSITY

Garbling an AND gate with Yao



u	v	w
0	0	0
0	1	0
1	0	0
1	1	1

Garbling an AND gate with Yao



$$E_{A_0,B_0}(C_0)$$
$$E_{A_0,B_1}(C_0)$$
$$E_{A_1,B_0}(C_0)$$
$$E_{A_1,B_1}(C_1)$$

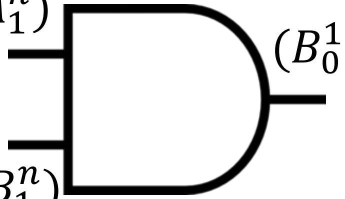
- Pick two random keys for each wire
- Encrypt the truth table of each gate

- Randomly **permute** entries
- **Invariant**: evaluator learns **one** key per wire throughout the circuit

Multi-party garbled circuits

[Beaver Micali Rogaway90]

$(A_0^1, \dots, A_0^n), (A_1^1, \dots, A_1^n)$



$(B_0^1, \dots, B_0^n), (B_1^1, \dots, B_1^n)$

$(B_0^1, \dots, B_0^n), (B_1^1, \dots, B_1^n)$

$E_{A_0,B_0}(C_0)$

$E_{A_0,B_1}(C_0)$

$E_{A_1,B_0}(C_0)$

$E_{A_1,B_1}(C_1)$

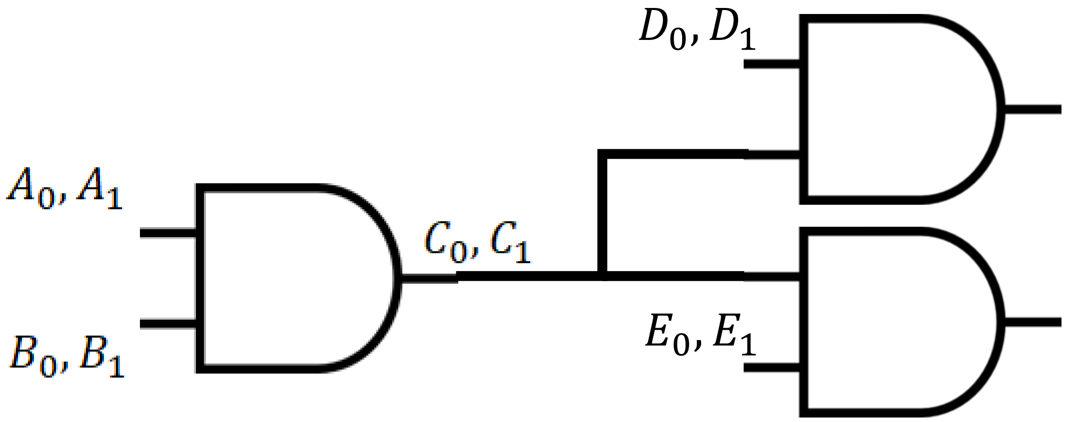
Shrink the keys!

Each P_i gets $A_0^i, A_1^i \in \{0,1\}^{\cancel{k}\,\ell}$ etc

Use distributed encryption: $E_{A,B}(C) = H(1 \,||\, A^1 || B^1) \oplus \dots \oplus H(n \,|| A^n || B^n) \oplus (C^1, \dots, C^n)$

For hash function $H : \{0,1\}^* \to \{0,1\}^{\cancel{nk}\, n\ell}$

BMR with short keys: a few technical challenges



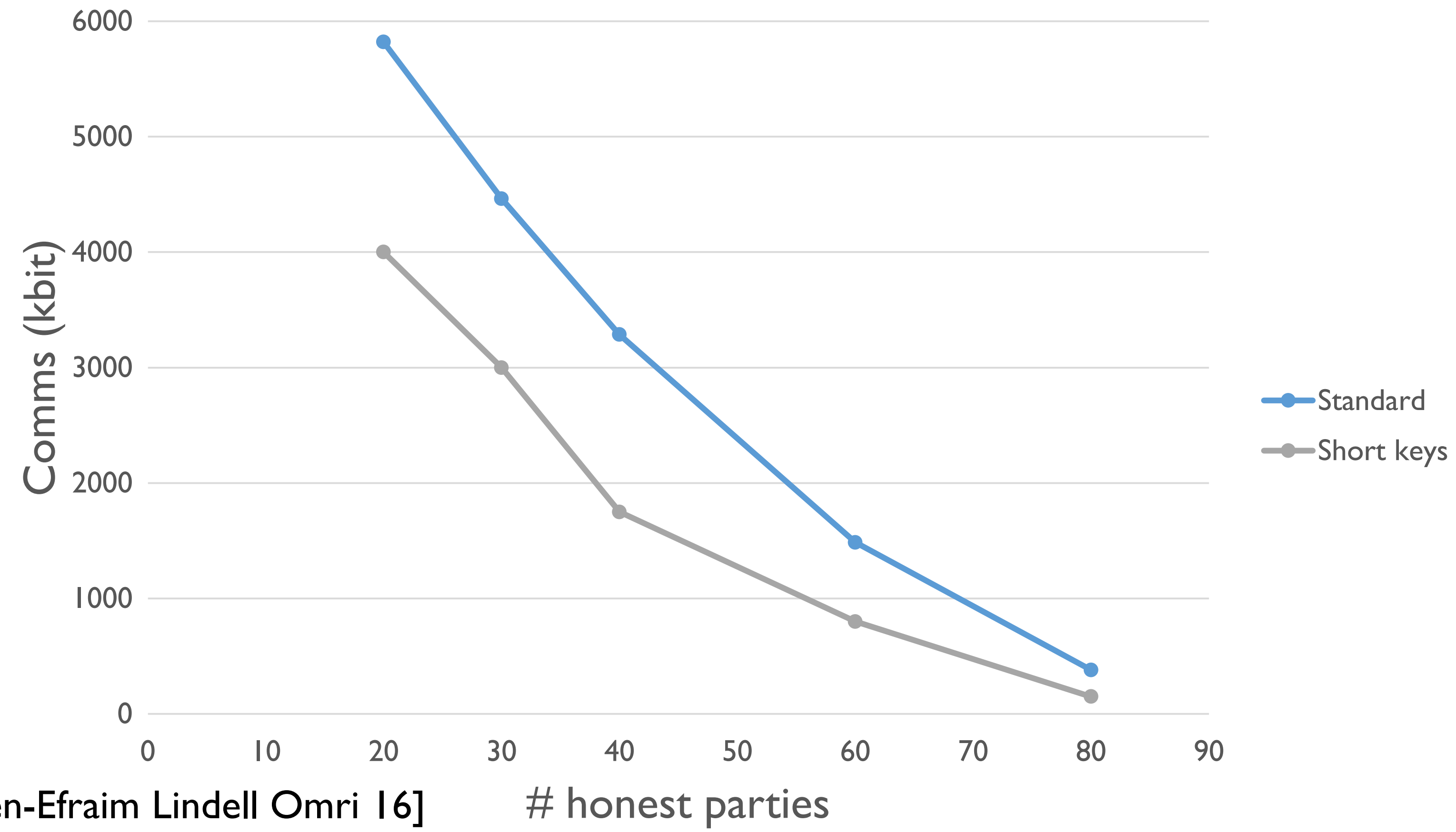
- Reusing keys reduces security in regular syndrome decoding
- Problem for:
  - High fan-out
  - Free-xor
- Solution:
  - Splitter gates [Tate Xu 03] – can be garbled for free
  - Local free-XOR offsets

BMR with short keys: pros and cons

- Garbled AND gate:
  - $4n\ell + 1$ bits vs $4nk$ bits previously
  - ℓ as small as 8
- Preprocessing phase:
  - Less communication using short keys
- Online phase:
  - $O(\frac{n^2\ell}{k})$ hash evaluations per garbled gate, vs $O(n^2)$ previously*
  - Need splitter gates: ≈1 splitter per (XOR/AND) gate

*or $O(1)$ using DDH/LWE [Ben-Efraim Lindell Omri 17]

Communication cost of garbling an AND gate (200 parties)



Comparison with [Ben-Efraim Lindell Omri 16]

Conclusion and future directions

- New technique for distributing trust in MPC
- More efficient protocols for 20+ parties
  - Also helps large-scale protocols with random committees

Future challenges:

- Active security
  - Information-theoretic MACs with short keys
- Arithmetic circuits
- Adaptive security
- Optimizations, cryptanalysis