# Multi-Party Computation of **Polynomials & Branching Programs** without Simultaneous Interaction


Hoeteck Wee (ENS)

\+

Dov Gordon (ACS)
Tal Malkin (Columbia)
Mike Rosulek (OSU)

# multi-party computation

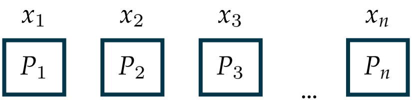
| $x_1$ | $x_2$ | $x_3$ | | $x_n$ |
|---|---|---|---|---|
| $P_1$ | $P_2$ | $P_3$ | ... | $P_n$ |

$$f(x_1, x_2, \ldots, x_n)$$

electronic voting

secure auctions

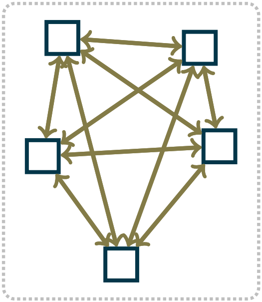
# multi-party computation



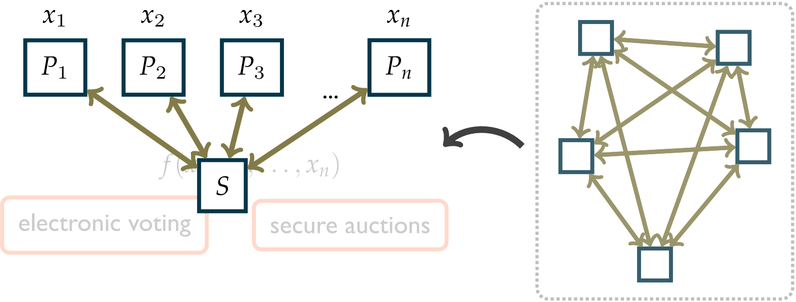

$$f(x_1, x_2, \ldots, x_n)$$

electronic voting

secure auctions



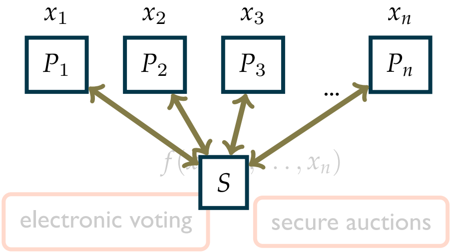
# multi-party computation on the web




**limited interaction** e.g. web users, program committees

[Ibrahim Kiayias Yung Zhou 09, Halevi Lindell Pinkas 11]

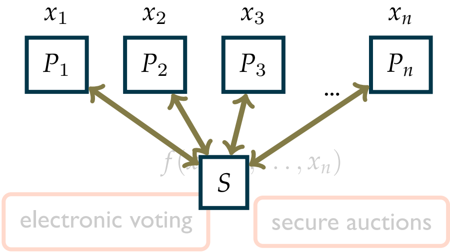
# multi-party computation on the web




**"one-pass" secure computation.** [Halevi Lindell Pinkas 11]

- each party interacts once with server in fixed order
- server announces result

# multi-party computation on the web



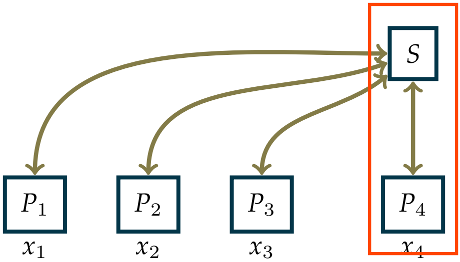

**"one-pass" secure computation.** [Halevi Lindell Pinkas 11]

- each party interacts once with server in fixed order
- server announces result
- server may be corrupt and colluding with parties

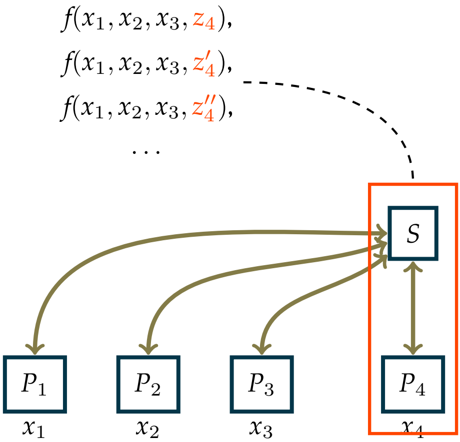
$\Rightarrow$ new technical challenge beyond standard MPC

# security: inherent leakage

$S$ colludes with last $k$ parties:



# security: inherent leakage

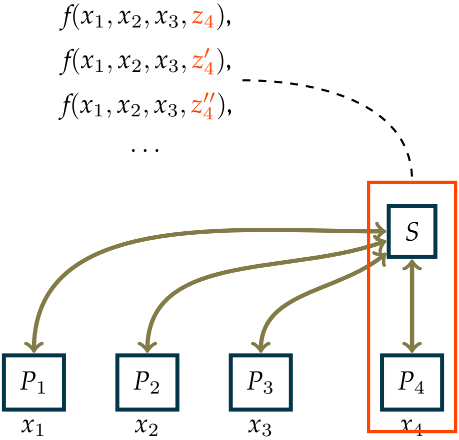



$S$ colludes with last $k$ parties:

Repeatedly:

- run protocol on choice of $z_{n-k+1}, \ldots, z_n$
- learn $f(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)$

# security: inherent leakage




$S$ colludes with last $k$ parties:

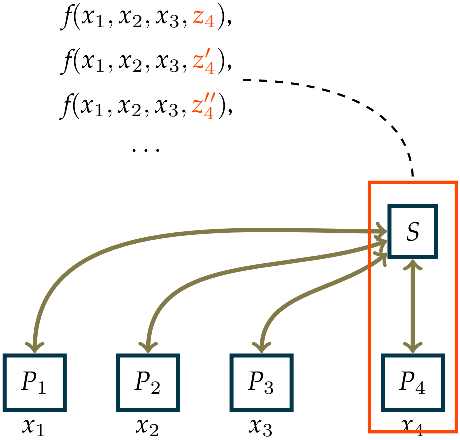
Repeatedly:

- run protocol on choice of $z_{n-k+1}, \ldots, z_n$
- learn $f(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)$

standard: single evaluation of $f$

here: multiple evaluations of $f$

# security: inherent leakage




$S$ colludes with last $k$ parties:

$\Rightarrow$ adversary gets oracle

$f(x_1, \dots, x_{n-k}, \star)$

Repeatedly:

- run protocol on choice of
  $z_{n-k+1}, \dots, z_n$
- learn
  $f(x_1, \dots, x_{n-k}, z_{n-k+1}, \dots, z_n)$

standard: single evaluation of $f$

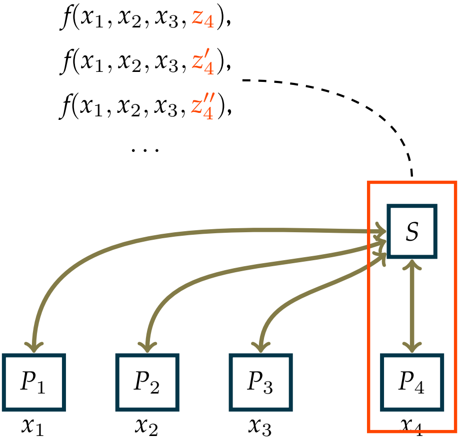
here: multiple evaluations of $f$

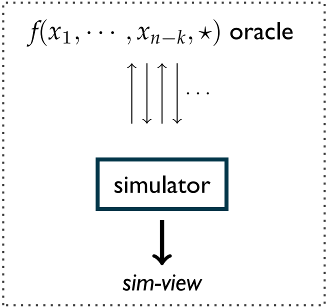
# security: inherent leakage

$S$ colludes with last $k$ parties:

$\Rightarrow$ adversary gets oracle

$$f(x_1, \ldots, x_{n-k}, \star)$$



# previous work

**Q.** what can we compute with secure, one-pass protocols? [HLP11]

- ✓ sum, selection, symmetric functions e.g. majority

  (via practical protocols)

- × pseudo-random functions

# previous work

**Q.** what can we compute with secure, one-pass protocols? [HLP11]

✓ sum, selection, symmetric functions e.g. majority

(via practical protocols)

✗ pseudo-random functions

**NB.** similar models, but no inherent leakage

— more than one pass [SYY99, IKOPS01, AJLTVW12]

— non-colluding server [IKYZ09]

# previous work

**Q.** what can we compute with secure, one-pass protocols? [HLP11]

✓ sum, selection, symmetric functions e.g. majority

(via practical protocols)

× pseudo-random functions

**NB.** related techniques, different context [IP07, HIK07]

# this work

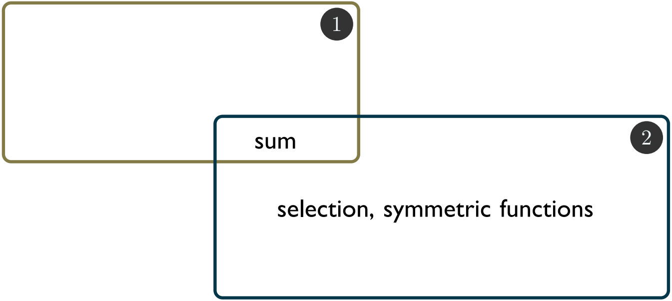
**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials (DCR)
2. read-once branching programs (DCR, DDH/DLIN, ...)

# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs



# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

1: low-degree polynomials

e.g. variance

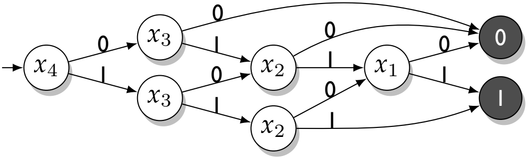
2: string matching, finite automata,

classification, second-price auction

# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

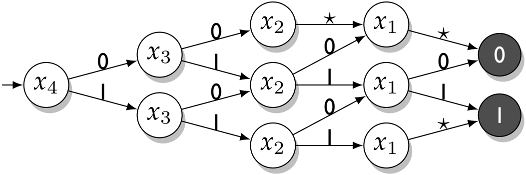



*"are at least $3$ of $\{x_1, \ldots, x_4\}$ equal to $1$?"*

# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

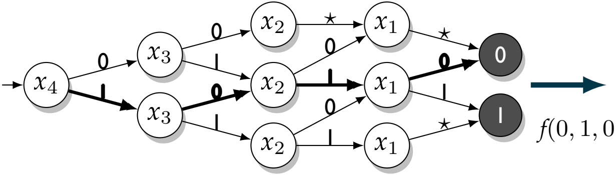



*"are at least* $3$ *of* $\{x_1, \ldots, x_4\}$ *equal to* $1$*?"*

# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs



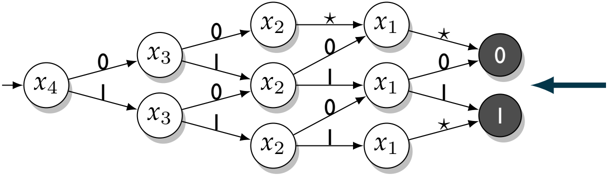

$f(0,1,0,1) = 0$

*"are at least* $3$ *of* $\{x_1, \ldots, x_4\}$ *equal to* $1$*?"*

# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs



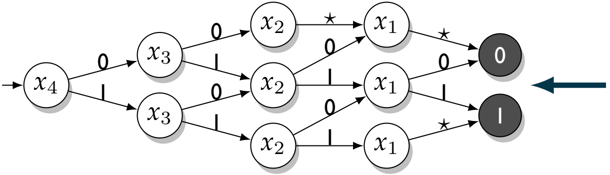

**our protocol.** in public key model

– right-to-left [IP07] + nested encryption [HLP11]

# this work

**theorem.** secure one-pass protocols for

1. sparse multi-variate polynomials
2. read-once branching programs

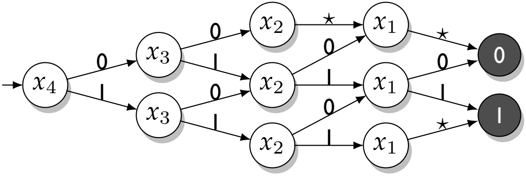



**our protocol.** in public key model

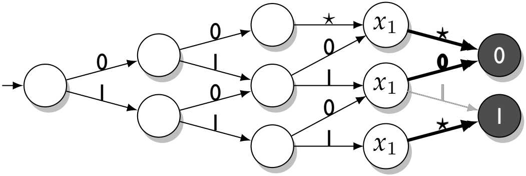
– right-to-left [IP07] + nested encryption [HLP11]

– this talk: honest-but-curious (malicious via NIZK / GS proofs)

# our protocol (warm-up)



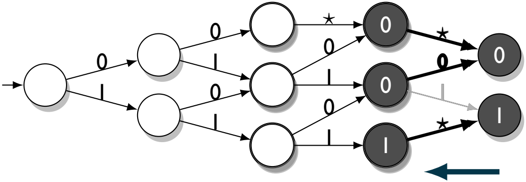
# our protocol (warm-up)




$P_1$

$x_1 = 0$

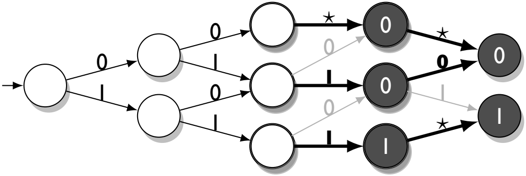
# our protocol (warm-up)




$P_1$

$x_1 = 0$

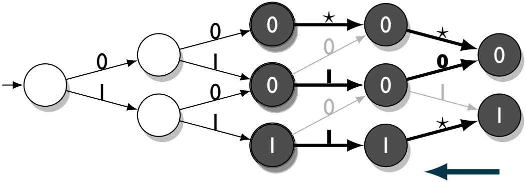
# our protocol (warm-up)





$x_1 = 0$ $x_2 = 1$

# our protocol (warm-up)



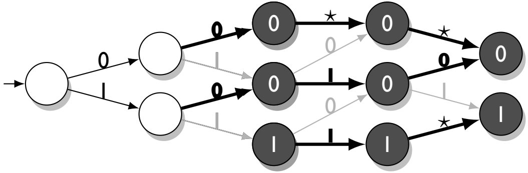

$P_1$

$x_1 = 0$

$P_2$

$x_2 = 1$

# our protocol (warm-up)

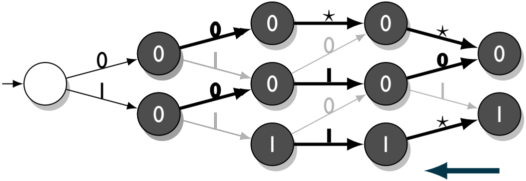



$P_1$ $x_1 = 0$

$P_2$ $x_2 = 1$

$P_3$ $x_3 = 0$

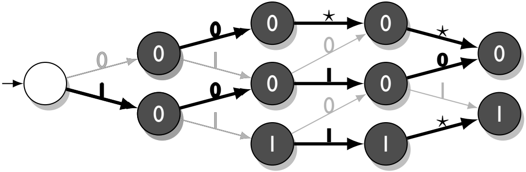
# our protocol (warm-up)



$P_1$ $P_2$ $P_3$

$x_1 = 0$ $x_2 = 1$ $x_3 = 0$

# our protocol (warm-up)



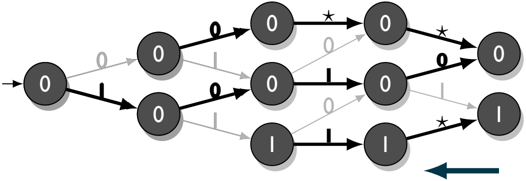

$P_1$ $x_1 = 0$

$P_2$ $x_2 = 1$

$P_3$ $x_3 = 0$

$P_4$ $x_4 = 1$

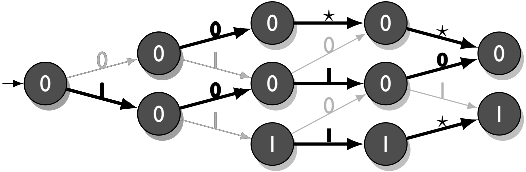
# our protocol (warm-up)




$P_1$ $P_2$ $P_3$ $P_4$

$x_1 = 0$ $x_2 = 1$ $x_3 = 0$ $x_4 = 1$

# our protocol (warm-up)



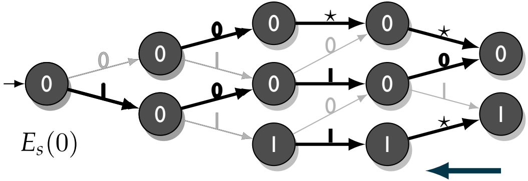
$E_1(E_2(E_3(E_4(E_s(0)))))$

$E_1(E_2(E_3(E_4(E_s(1)))))$

| $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|
| $x_1 = 0$ | $x_2 = 1$ | $x_3 = 0$ | $x_4 = 1$ |

**next.** propagate encrypted node labels “homomorphically”

# our protocol (warm-up)




$E_1(E_2(E_3(E_4(E_s(0)))))$

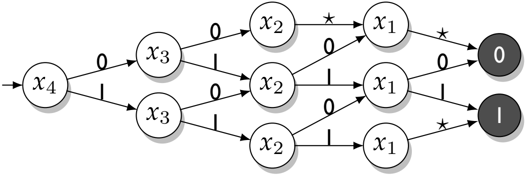
$E_1(E_2(E_3(E_4(E_s(1)))))$

$P_1$ $P_2$ $P_3$ $P_4$

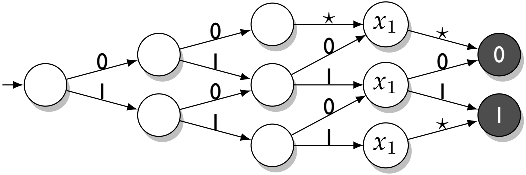
$x_1 = 0$ $x_2 = 1$ $x_3 = 0$ $x_4 = 1$

**next.** propagate encrypted node labels "homomorphically"

# our protocol



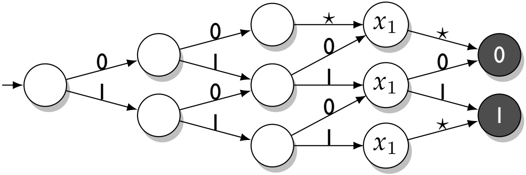
# our protocol





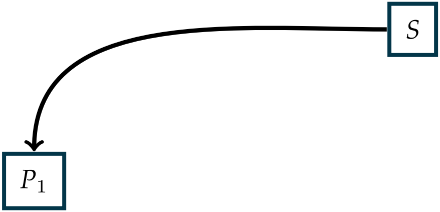
$x_1 = 0$

# our protocol



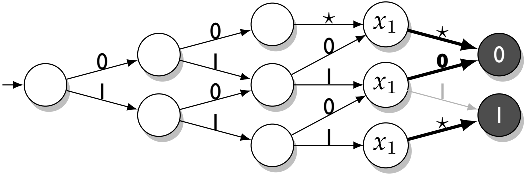

$E_1(E_2(E_3(E_4(E_s(0)))))$

$E_1(E_2(E_3(E_4(E_s(1)))))$




$x_1 = 0$

# our protocol




$E_1(E_2(E_3(E_4(E_s(0)))))$

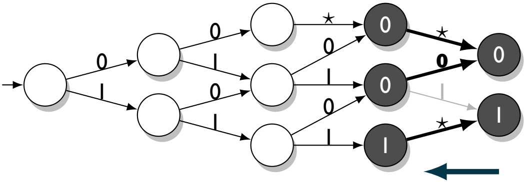
$E_1(E_2(E_3(E_4(E_s(1)))))$

$S$

$P_1$

$x_1 = 0$

# our protocol



$E_2(E_3(E_4(E_s(0))))$

$E_2(E_3(E_4(E_s(0))))$

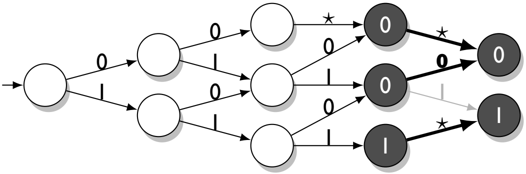
$E_2(E_3(E_4(E_s(1))))$

$S$

$P_1$

$x_1 = 0$

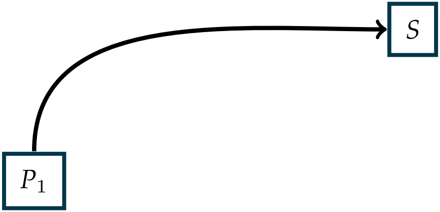
# our protocol




$$E_2(E_3(E_4(E_s(0))))$$

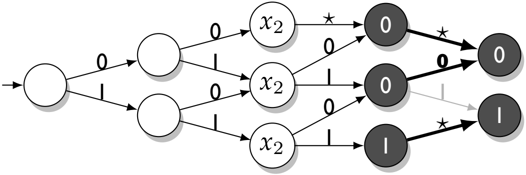
$$E_2(E_3(E_4(E_s(0))))$$

$$E_2(E_3(E_4(E_s(1))))$$




$x_1 = 0$

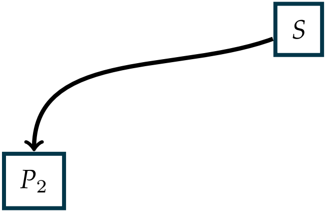
# our protocol




$E_2(E_3(E_4(E_s(0))))$

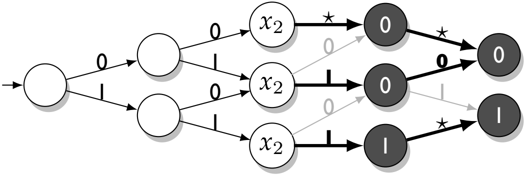
$E_2(E_3(E_4(E_s(0))))$

$E_2(E_3(E_4(E_s(1))))$




$x_2 = 1$

# our protocol




$E_2(E_3(E_4(E_s(0))))$

$E_2(E_3(E_4(E_s(0))))$

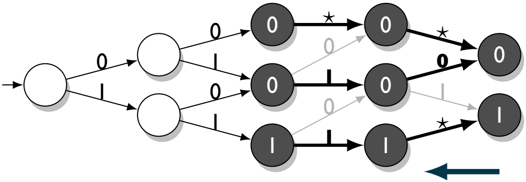
$E_2(E_3(E_4(E_s(1))))$

$S$

$P_2$

$x_2 = 1$

# our protocol



$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(0)))$

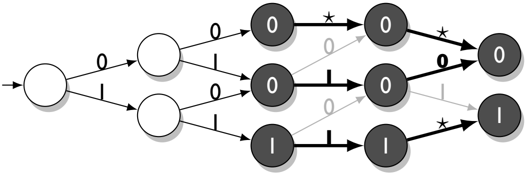
$E_3(E_4(E_s(1)))$

$S$

$P_2$

$x_2 = 1$

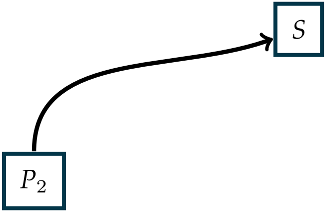
# our protocol




$E_3(E_4(E_s(0)))$

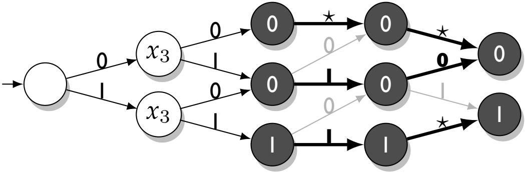
$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(1)))$




$x_2 = 1$

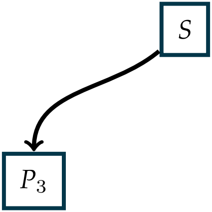
# our protocol




$E_3(E_4(E_s(0)))$

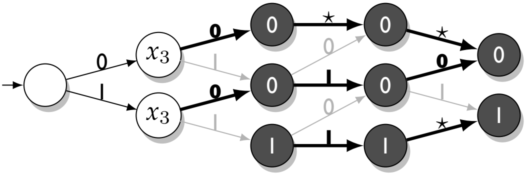
$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(1)))$




$x_3 = 0$

# our protocol




$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(0)))$

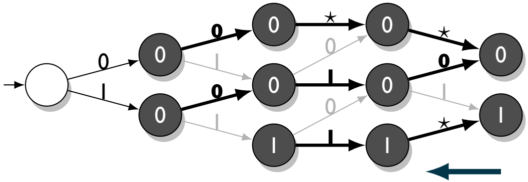
$E_3(E_4(E_s(1)))$

$S$

$P_3$

$x_3 = 0$

# our protocol



$E_4(E_s(0))$

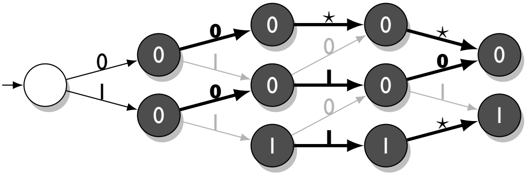
$E_4(E_s(0))$

$S$

$P_3$

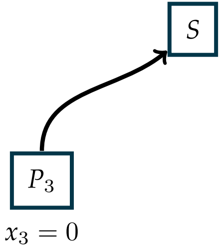
$x_3 = 0$

# our protocol

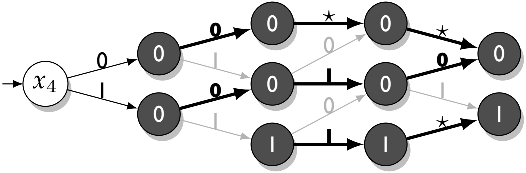


$E_4(E_s(0))$

$E_4(E_s(0))$

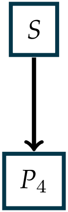


# our protocol



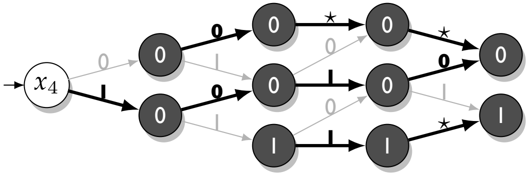

$E_4(E_s(0))$

$E_4(E_s(0))$




$x_4 = 1$

# our protocol




$E_4(E_s(0))$

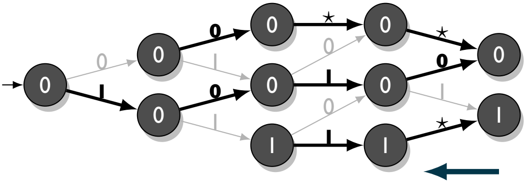
$E_4(E_s(0))$

$S$

$P_4$

$x_4 = 1$

# our protocol



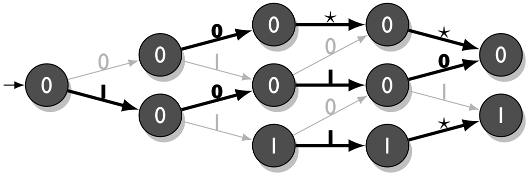

$E_s(0)$

$S$

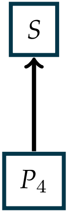
$P_4$

$x_4 = 1$

# our protocol

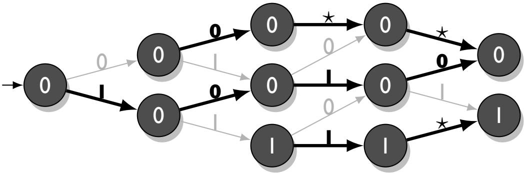


$E_s(0)$




$x_4 = 1$

# our protocol

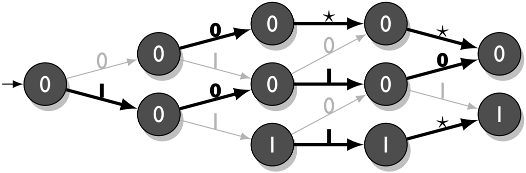


$E_s(0)$

*result = 0*

$S$

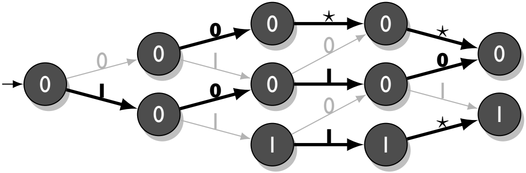
# our protocol




$E_s(0)$

**efficiency.** $O(\text{width})$ exponentiations per player under DCR, DDH/DLIN, ...

# our protocol

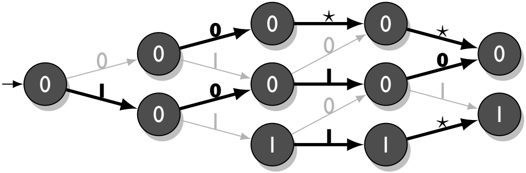


$E_s(0)$

**efficiency.** $O(\text{width})$ exponentiations per player under DCR, DDH/DLIN, ...

**security I.** honest $S$ – all messages protected by $E_s(\cdot)$

# our protocol




$E_s(0)$

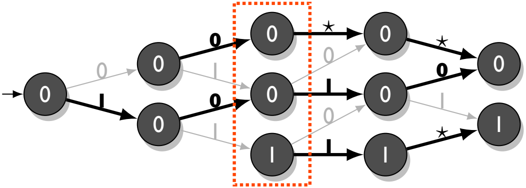
**efficiency.** $O(\text{width})$ exponentiations per player under DCR, DDH/DLIN, ...

**security I.** honest $S$ – all messages protected by $E_s(\cdot)$

**security II.** corrupt $S, P_3, P_4$ – need to simulate view given $f(x_1, x_2, \star)$

but not $x_1, x_2$

# our protocol



$$E_3(E_4(E_s(0)))$$
$$E_3(E_4(E_s(0)))$$
$$E_3(E_4(E_s(1)))$$

$P_2 \to S$

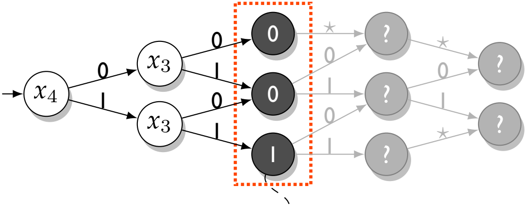
**efficiency.** $O(\text{width})$ exponentiations per player under DCR, DDH/DLIN, ...

**security I.** honest $S$ – all messages protected by $E_s(\cdot)$

**security II.** corrupt $S, P_3, P_4$ – need to simulate view given $f(x_1, x_2, \star)$

but not $x_1, x_2$

# our protocol



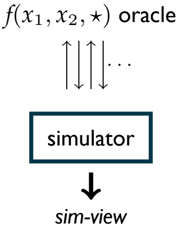

$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(0)))$

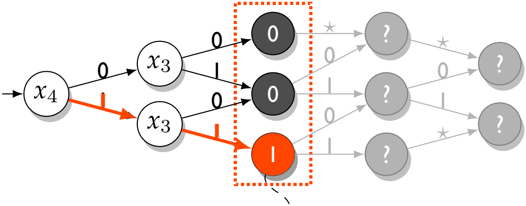
$E_3(E_4(E_s(1)))$

*"How to simulate these*

*node labels (unencrypted)?"*



# our protocol



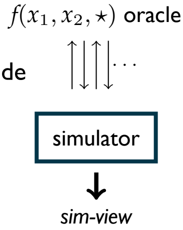

$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(0)))$

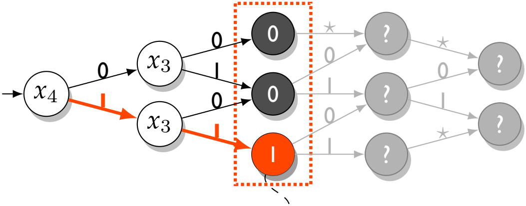
$E_3(E_4(E_s(1)))$

*"How to simulate these node labels (unencrypted)?"*

- for each node, use BFS to find a path from start node



# our protocol




$E_3(E_4(E_s(0)))$

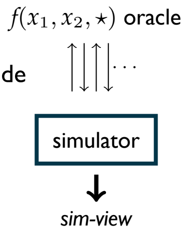
$E_3(E_4(E_s(0)))$

$E_3(E_4(E_s(1)))$

*"How to simulate these node labels (unencrypted)?"*

- for each node, use BFS to find a path from start node
- call oracle on inputs induced by path

  e.g. query $f(x_1, x_2, \star)$ on $(1, 1)$



# conclusion

**this work.** secure one-pass protocols

1. sparse multi-variate polynomials
2. read-once branching programs

**open questions.**

- larger classes, e.g. linear branching programs [HIK07]?
- impossibility results / complete characterization?
- better efficiency, e.g. second-price auctions?

the end