Randomness in Private Sequential Stateless Protocols

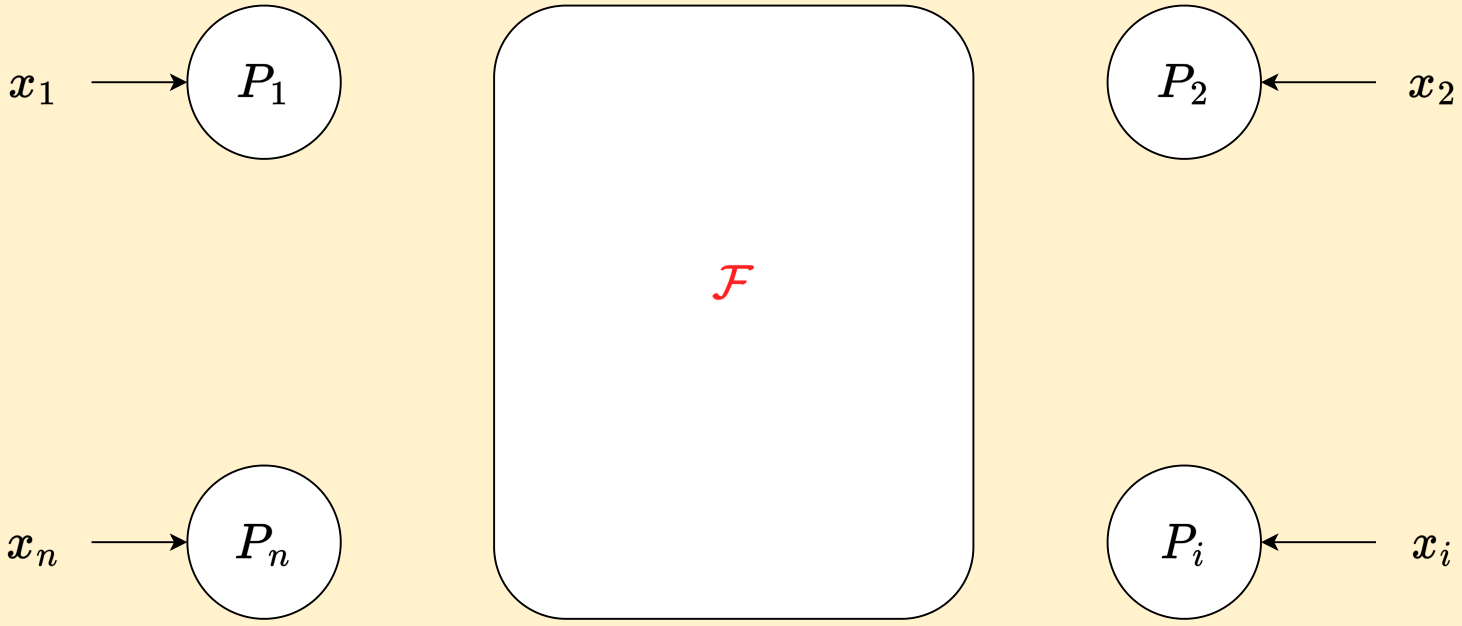
Hari Krishnan P A
TIFR

Varun Narayanan
UCLA

Manoj Prabhakaran
IIT Bombay

Vinod Prabhakaran
TIFR

Secure multi-party computation



No party should learn anything apart from their own input and output of the function.

Randomness in MPC

Randomness: Central resource in cryptography, especially in unconditional cryptography

- Randomness efficient computation (and lower bounds).
  - Kushilevitz, Mansour (PODC 1996),
  - Kushilevitz, Ostrovsky, Rosén (STOC 1996)
  - Canetti, Kushilevitz, Ostrovsky, Rosén (PODC 1997)
  - Gal, Rosén (STOC 2003)
  - Jakoby, Liskiewicz, Reischuk (STACS 2003)
  - Blundo, Galdi, Persiano (2007)
  - Kushilevitz, Ostrovsky, Prouff, Rosén, Thillard, Vergnaud (TCC - 2019)
  - Goyal, Ishai, Song (CRYPTO 2022)
  - Couteau, Rosén (Asiacrypt 2022)

Randomness in MPC

Kushilevitz, Ostrovsky, Rosén. (STOC 1996)

Randomness complexity of 1-private computation

Circuit complexity

1-privacy: Semi-honest corruption of any one party

Our Results

Randomness complexity of private sequential stateless (PSS) computation

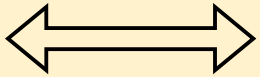
Branching program complexity

Our Results

Randomness complexity of private sequential stateless (PSS) computation

Branching program complexity

f has a speak-once PSS protocol with constant randomness



f has an read-once constant-width branching program

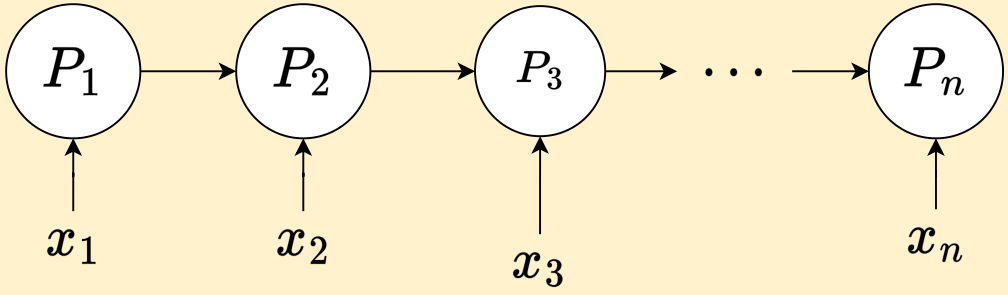
$$f : \{0,1\}^n \to \{0,1\}$$

Extends to speak-$O(k)$ PSS and read-$O(k)$ BP

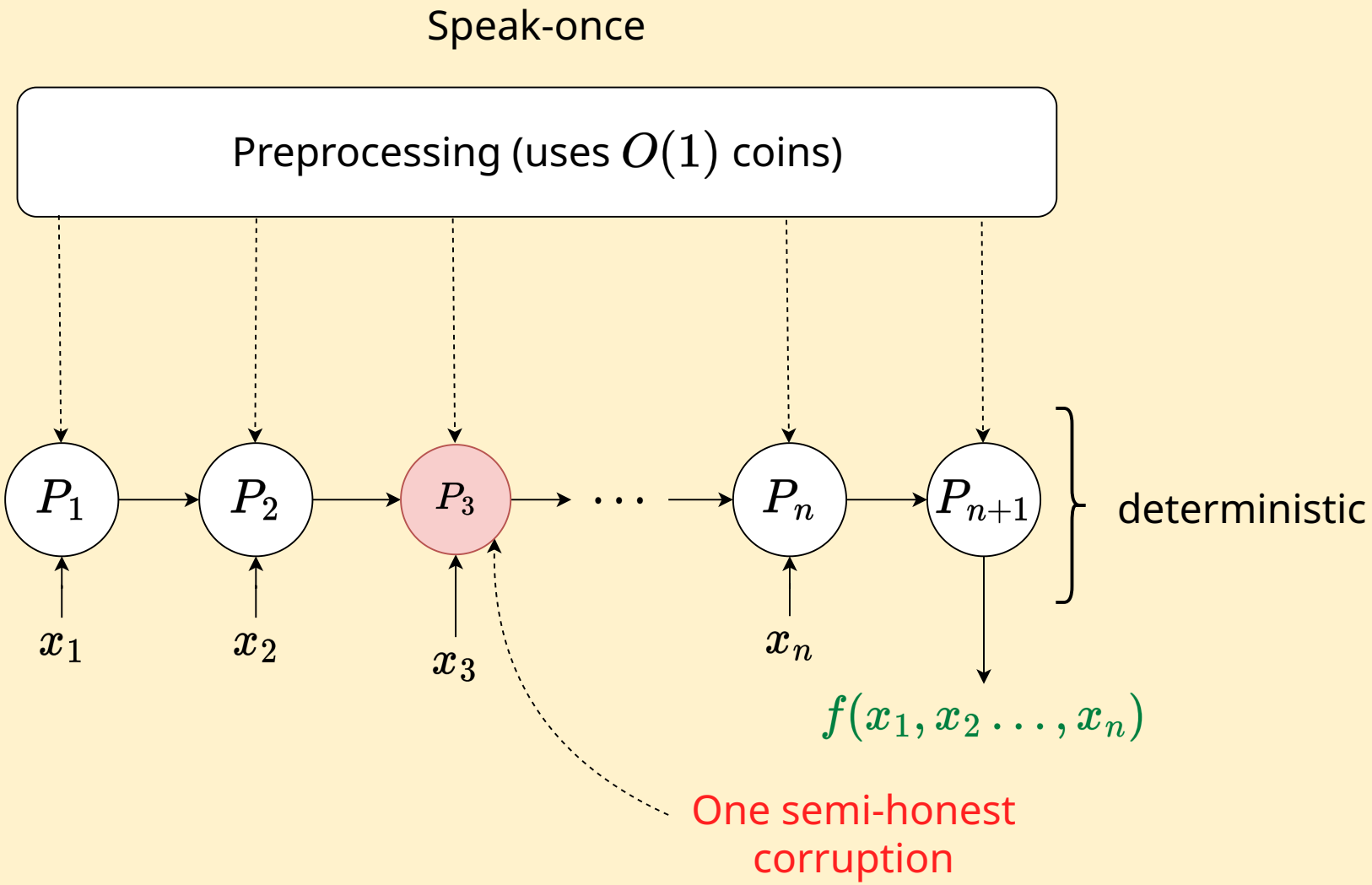
where k is independent of the input size n

Sequential Model

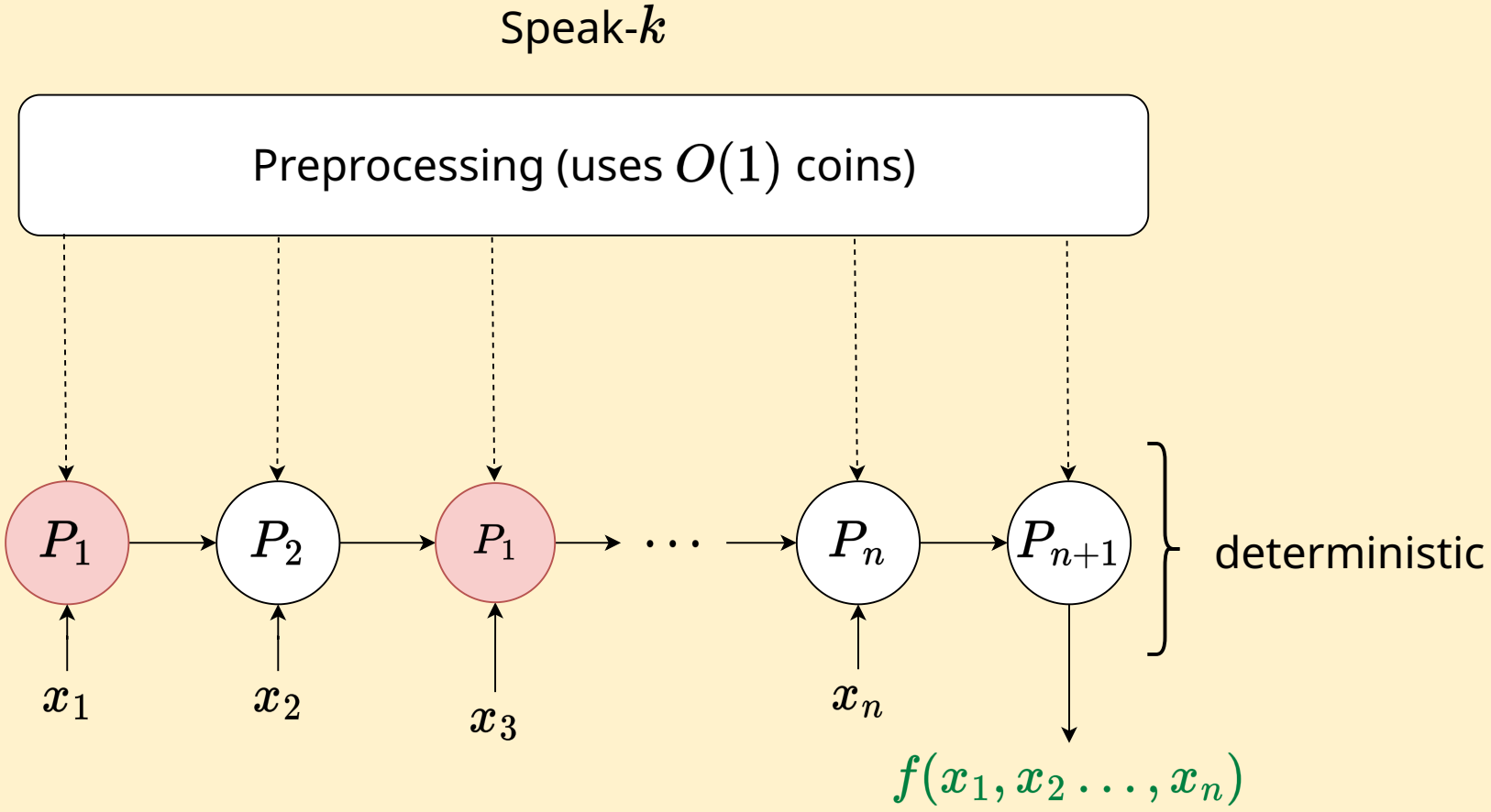
Speak-once



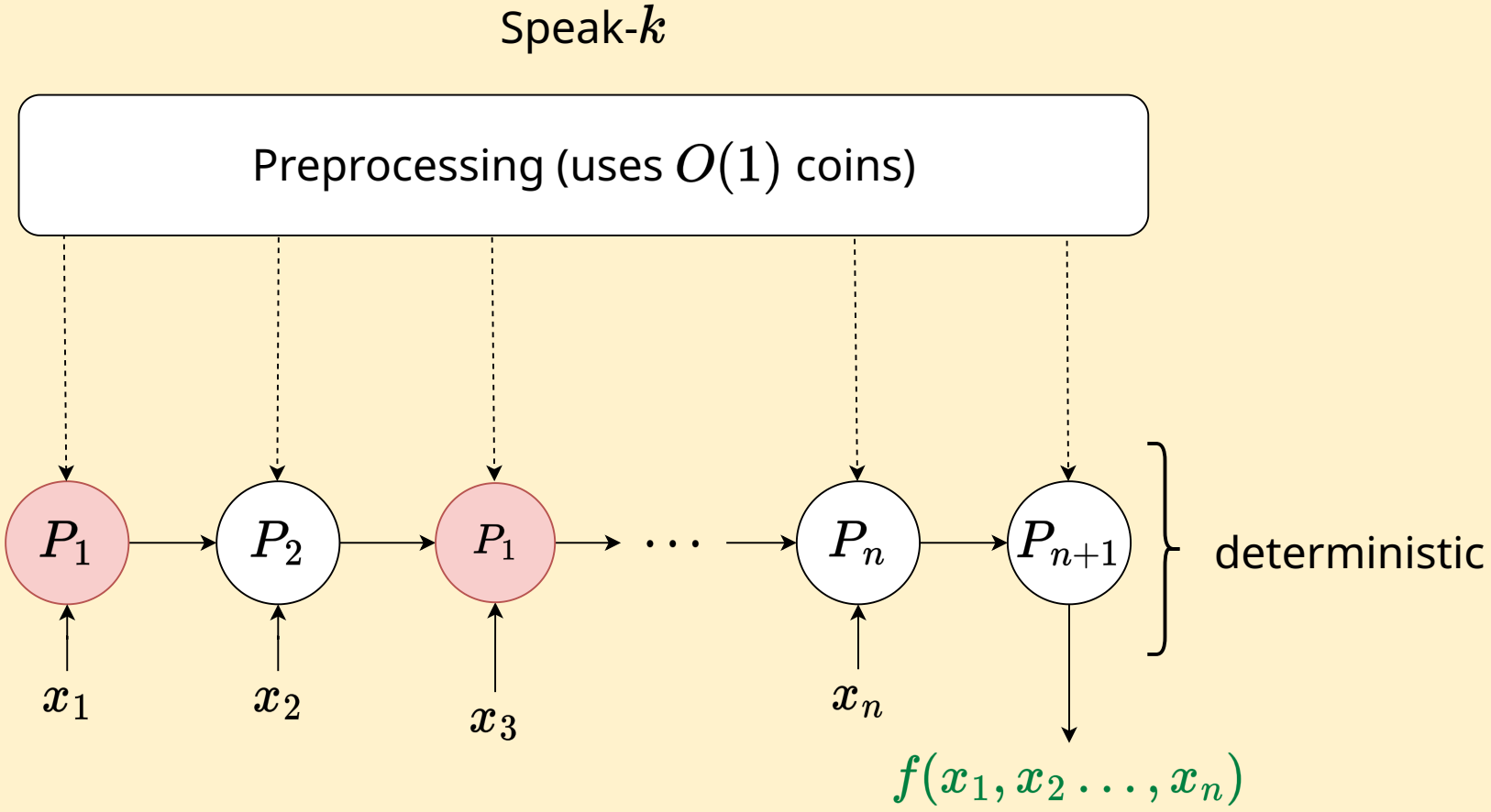
Sequential Model



Sequential Model



Sequential Model



Stateless: Parties do not maintain states between rounds when they speak

Motivation

Other simple models (having star topology)

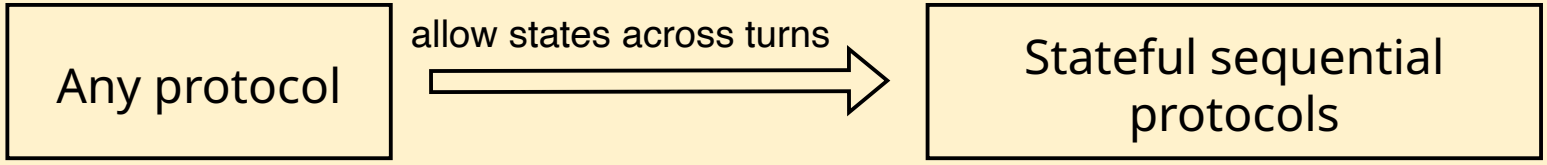
- Private simultaneous messaging
- Non-interactive secure computation
- Conditional disclosure of secrets

Motivation

Other simple models (having star topology)

- Private simultaneous messaging
- Non-interactive secure computation
- Conditional disclosure of secrets

Why stateless?

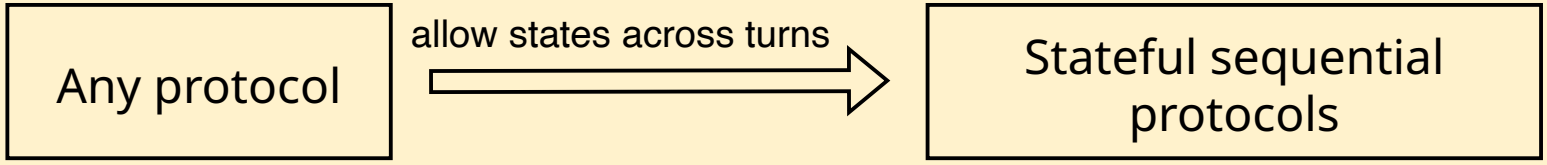


Motivation

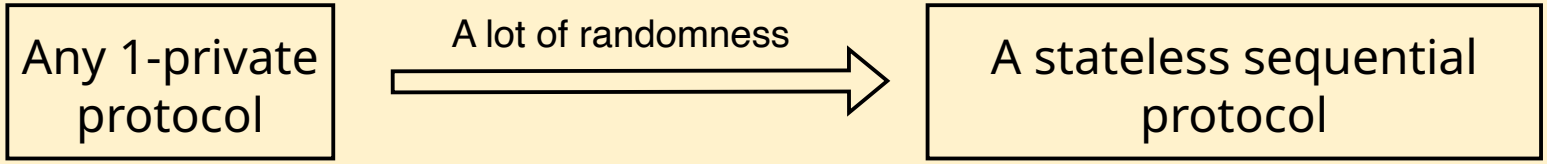
Other simple models (having star topology)

- Private simultaneous messaging
- Non-interactive secure computation
- Conditional disclosure of secrets

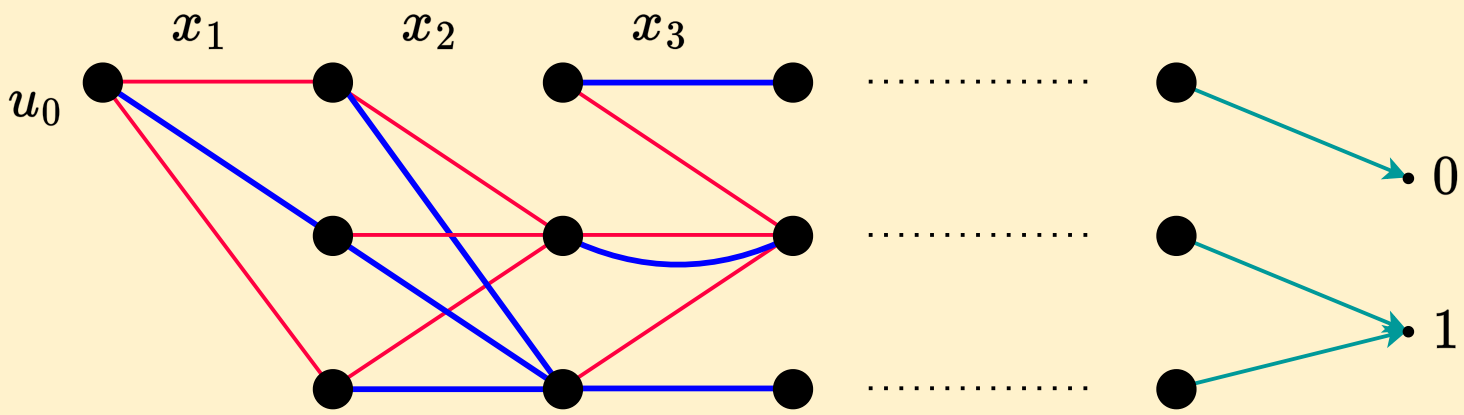
Why stateless?



Why is restricting randomness interesting



Read-once branching programs



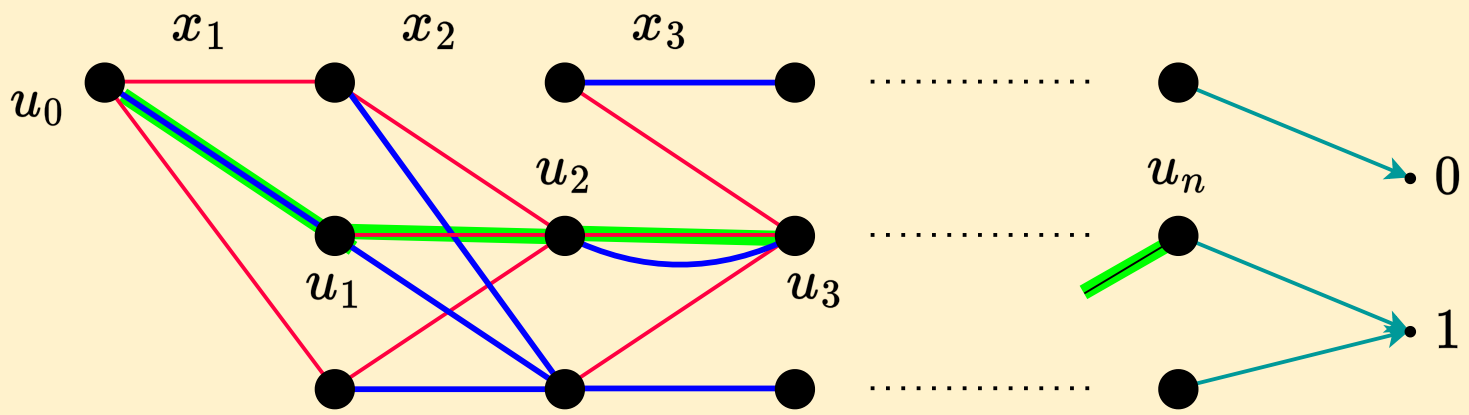
$\mathtt{Tr}_0^{(1)}$ $\mathtt{Tr}_0^{(2)}$ $\mathtt{Tr}_0^{(3)}$

$\mathtt{Tr}_1^{(1)}$ $\mathtt{Tr}_1^{(2)}$ $\mathtt{Tr}_1^{(3)}$

φ

Width $w = 3$ here

Read-once branching programs



$\mathtt{Tr}_0^{(1)}$ $\mathtt{Tr}_0^{(2)}$ $\mathtt{Tr}_0^{(3)}$

$\mathtt{Tr}_1^{(1)}$ $\mathtt{Tr}_1^{(2)}$ $\mathtt{Tr}_1^{(3)}$

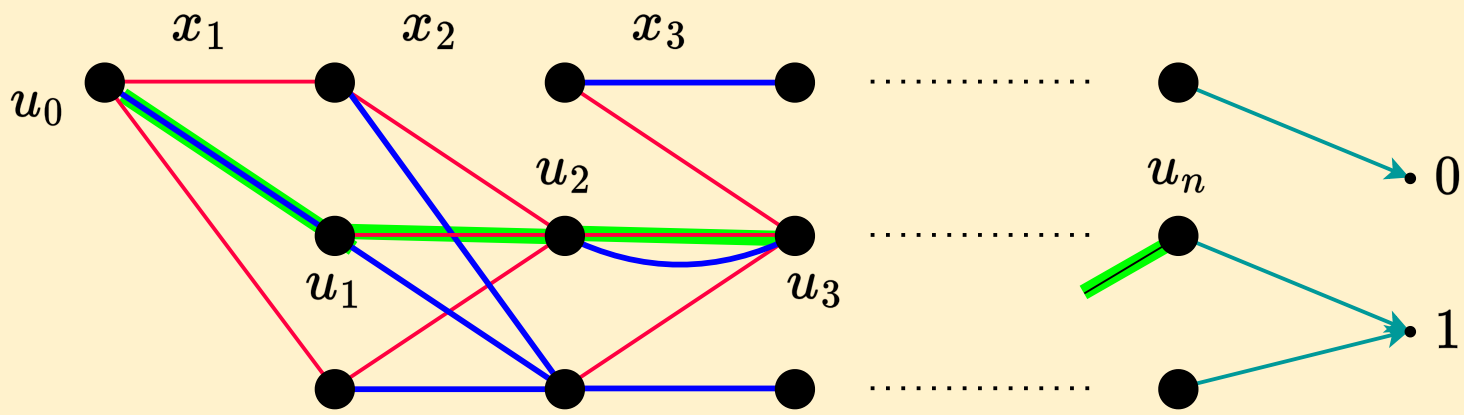
φ

Width $w = 3$ here

If $x_i = 0$ $\quad u_{i+1} = \mathtt{Tr}_0^{(i)}(u_i)$

If $x_i = 1$ $\quad u_{i+1} = \mathtt{Tr}_1^{(i)}(u_i)$

Read-once branching programs



$\mathtt{Tr}_0^{(1)}$ $\mathtt{Tr}_0^{(2)}$ $\mathtt{Tr}_0^{(3)}$

$\mathtt{Tr}_1^{(1)}$ $\mathtt{Tr}_1^{(2)}$ $\mathtt{Tr}_1^{(3)}$

φ

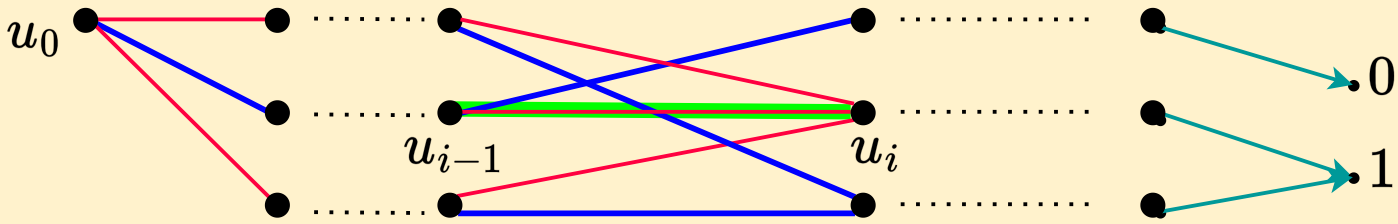
Width $w = 3$ here

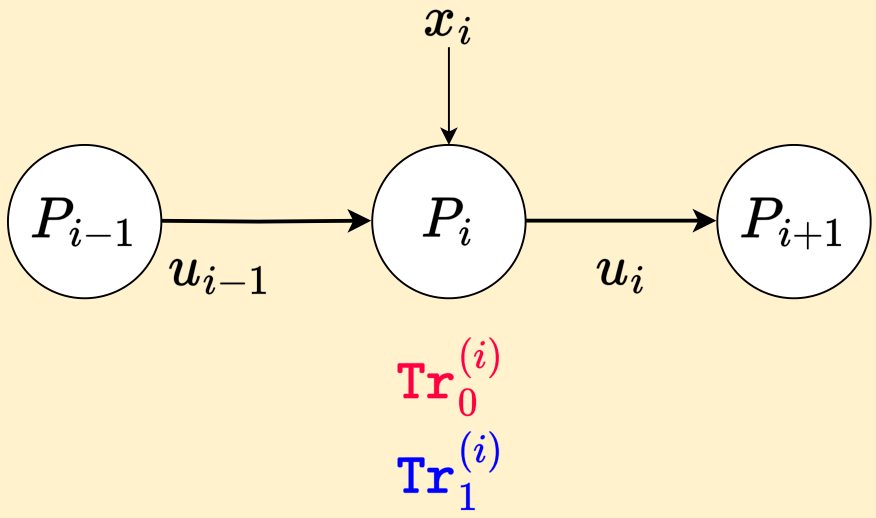
If $x_i = 0$ $\quad u_{i+1} = \mathtt{Tr}_0^{(i)}(u_i)$

If $x_i = 1$ $\quad u_{i+1} = \mathtt{Tr}_1^{(i)}(u_i)$

$$f(x_1, \ldots, x_n) = \varphi(u_n)$$

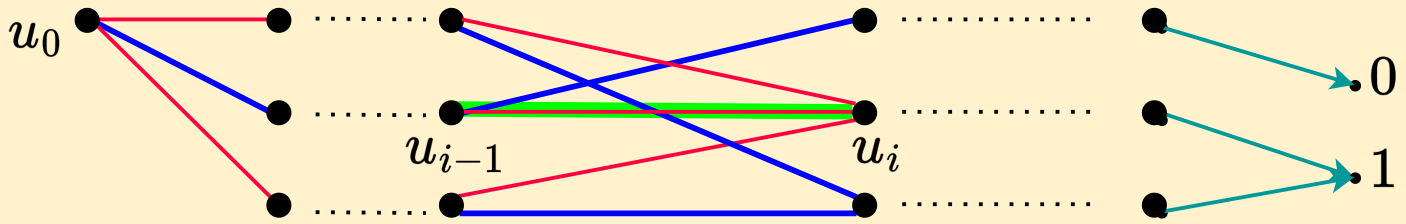
Protocol for evaluating a branching program insecurely



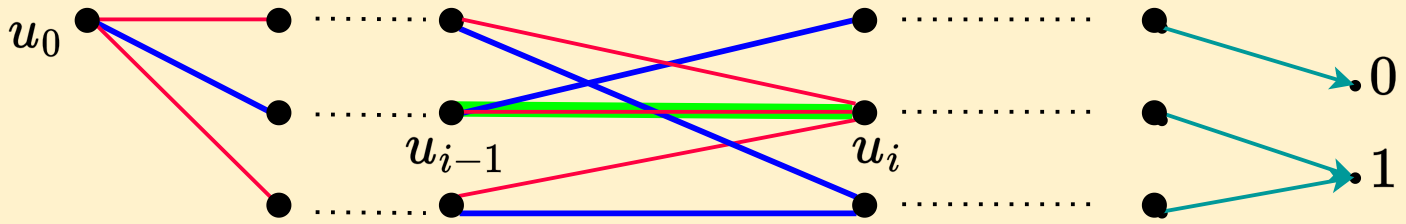


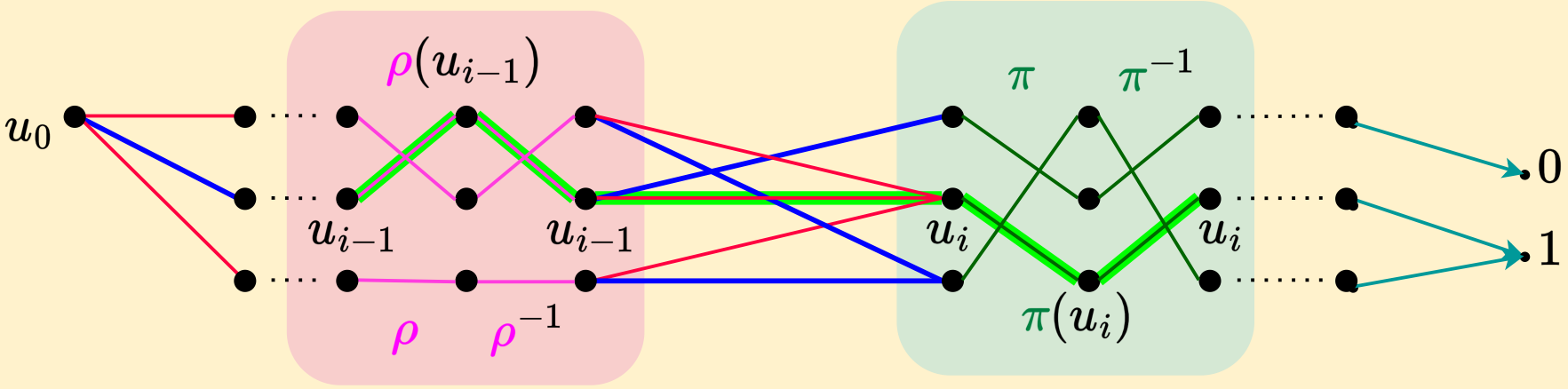
P_i receives u_{i-1}, which is insecure

Secure evaluation - Attempt 1



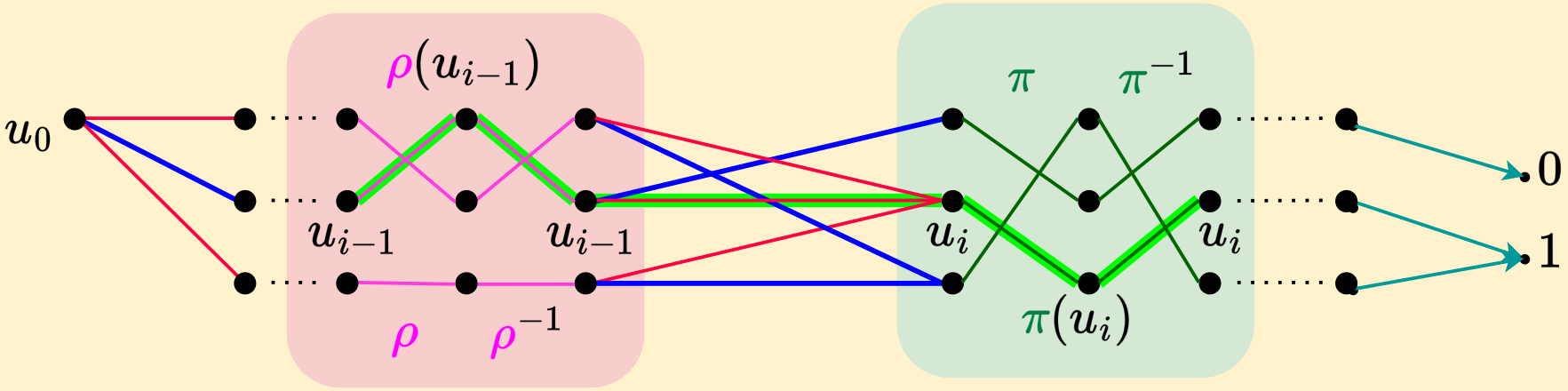
Secure evaluation - Attempt 1



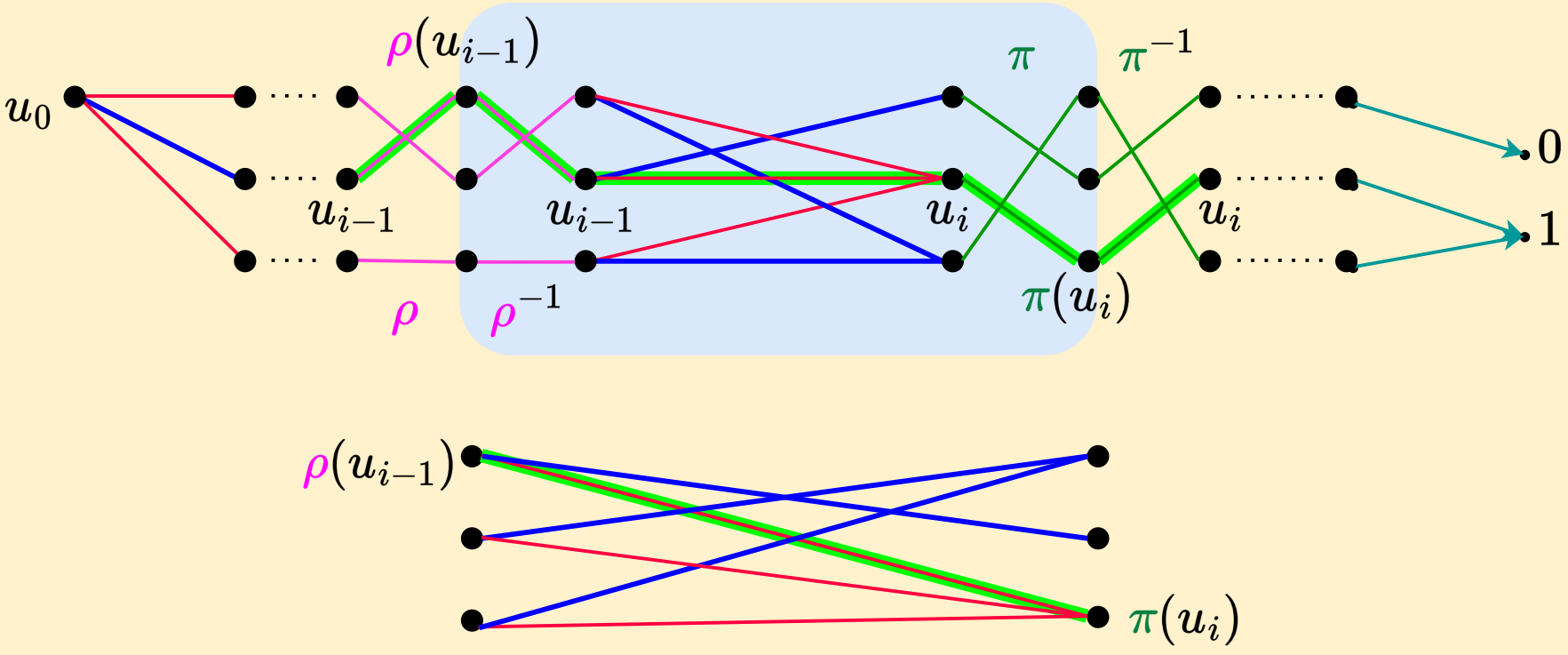


π, ρ are uniformly chosen permutations

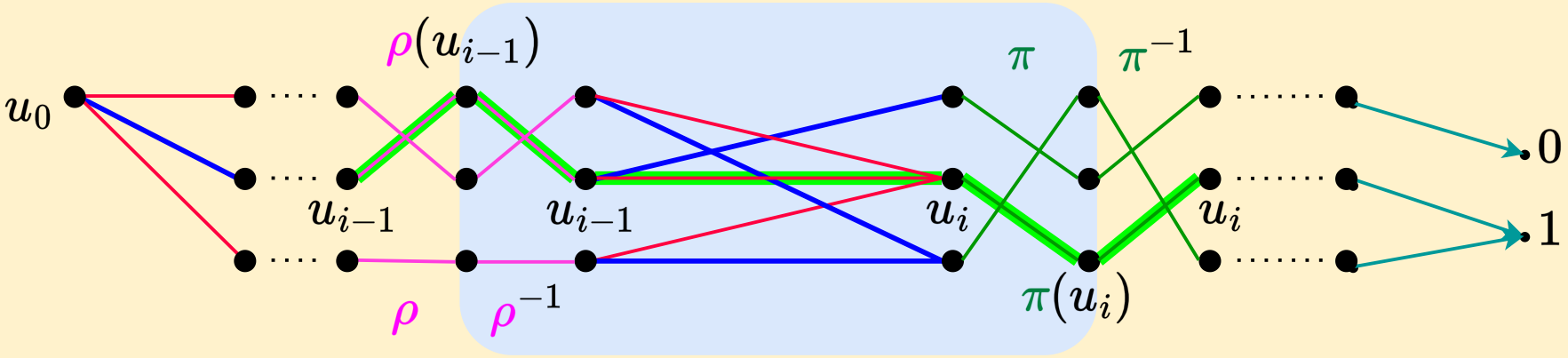
Secure evaluation - Attempt 1

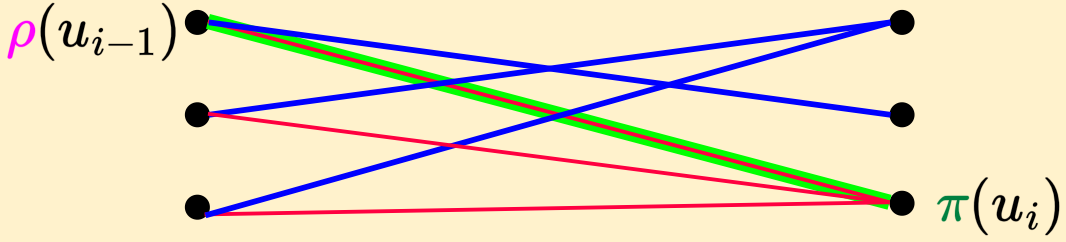


Secure evaluation - Attempt 1



Secure evaluation - Attempt 1

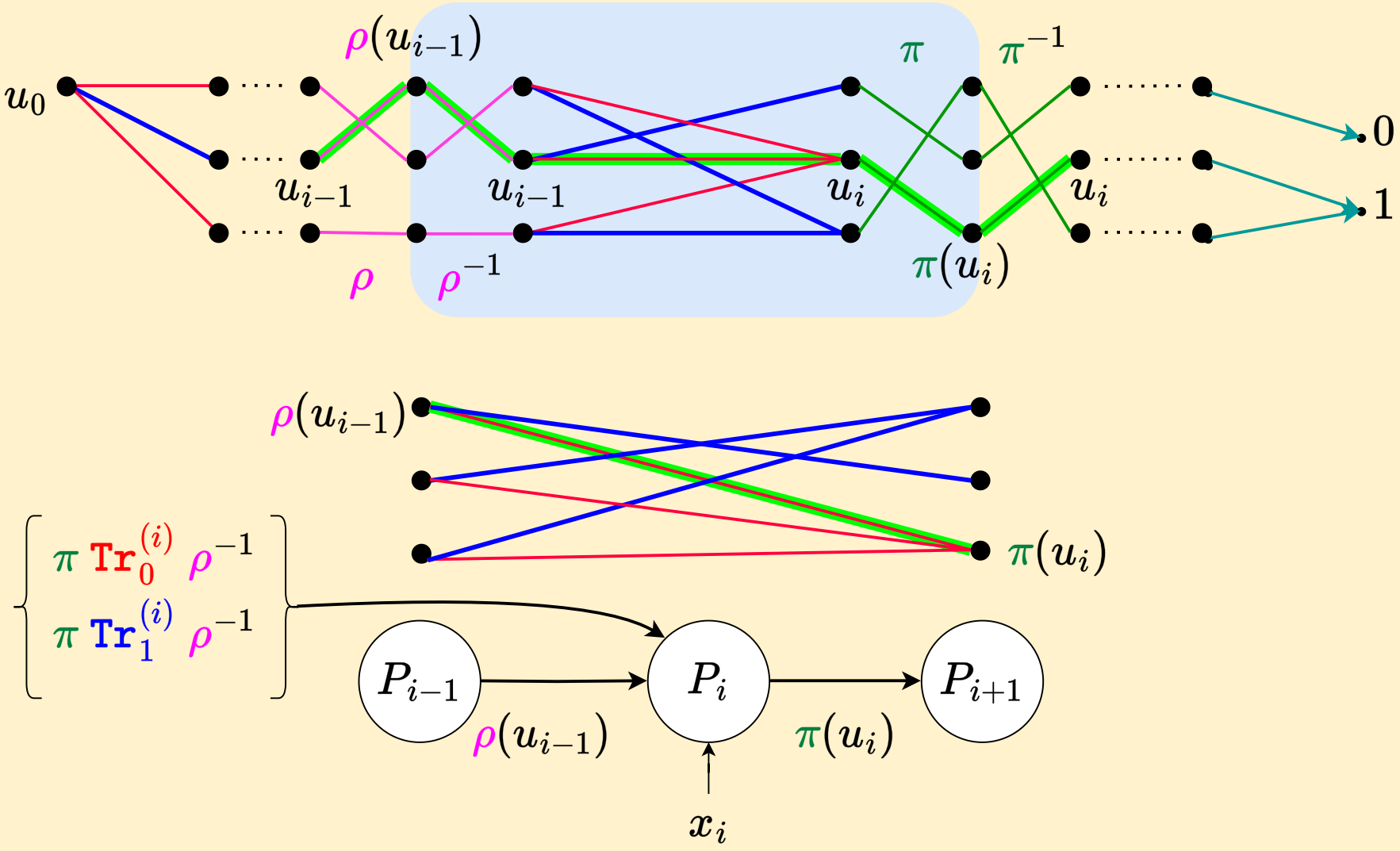




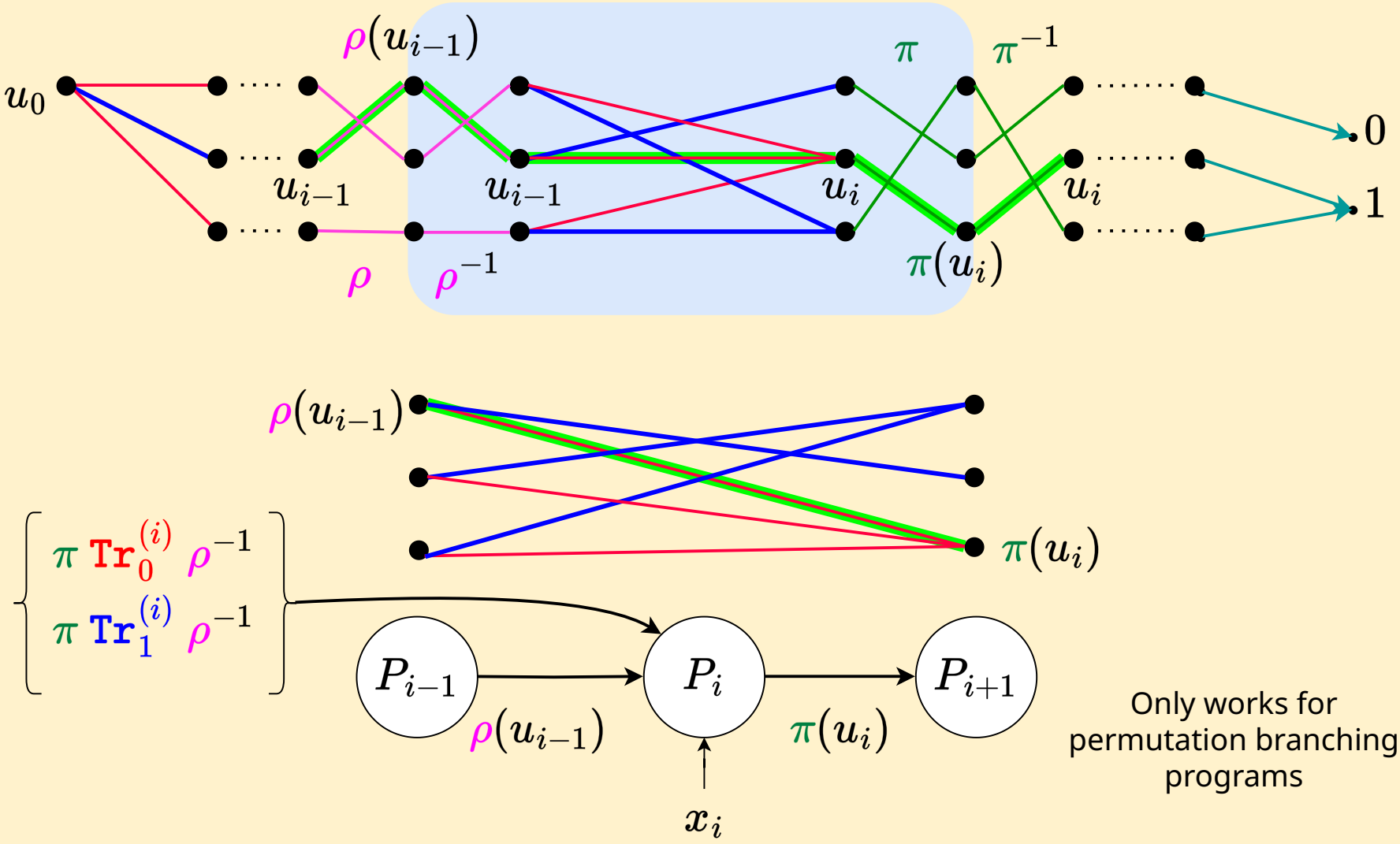
$\pi\ \mathtt{Tr}_0^{(i)}\ \rho^{-1}$

$\pi\ \mathtt{Tr}_1^{(i)}\ \rho^{-1}$

Secure evaluation - Attempt 1

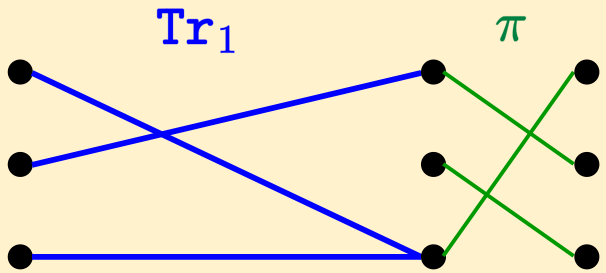


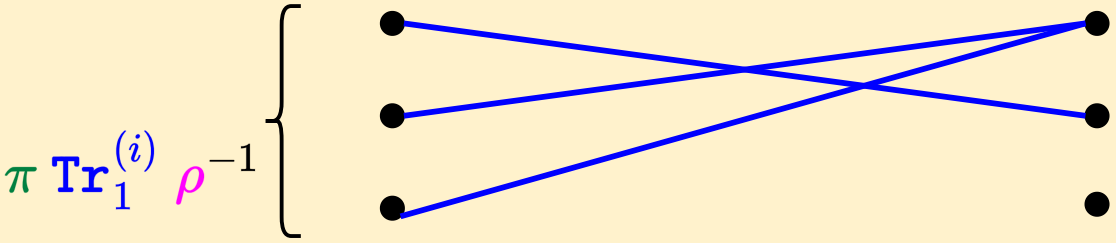
Secure evaluation - Attempt 1



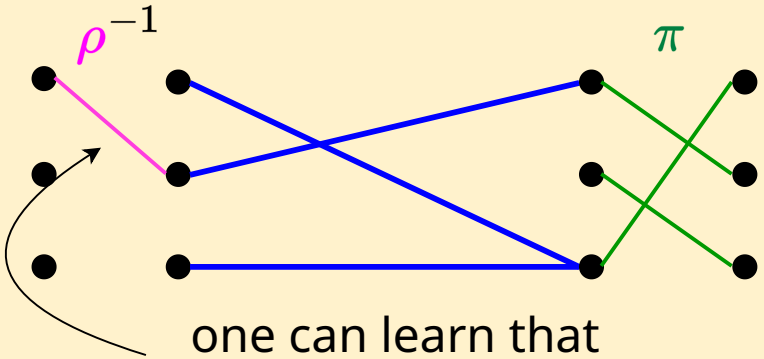
Only works for permutation branching programs

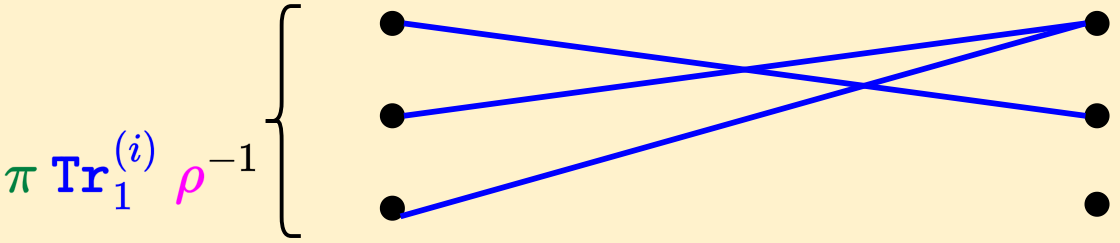
Secure evaluation - Attempt 1 fails most of the times

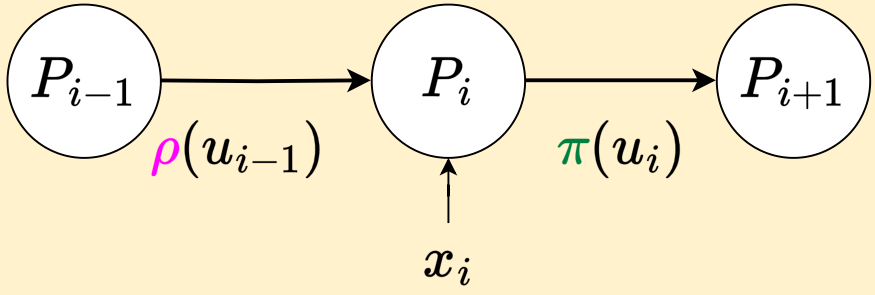




Secure evaluation - Attempt 1 fails most of the times







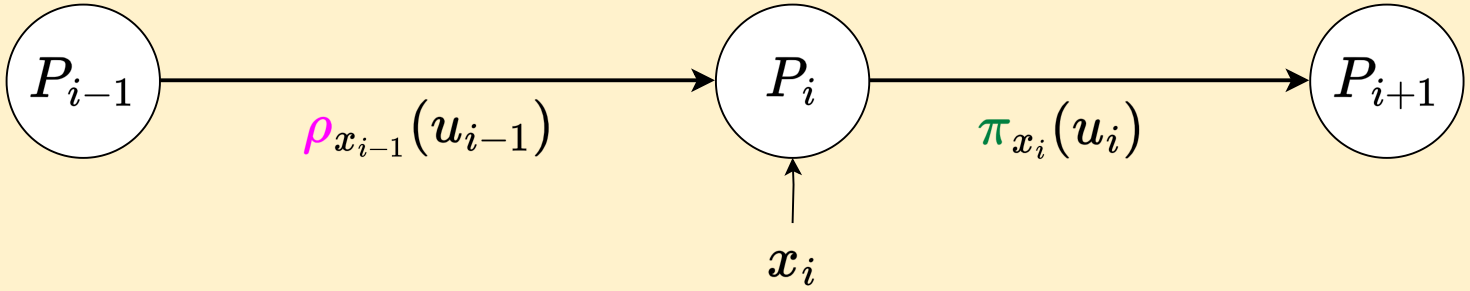
Secure evaluation - Attempt 2

- Workaround - Use two different sets of masks - π_0, π_1 instead of π
  ρ_0, ρ_1 instead of ρ

Secure evaluation - Attempt 2

- Workaround - Use two different sets of masks - π_0, π_1 instead of π
  ρ_0, ρ_1 instead of ρ

  P_i needs to map $\rho_{x_{i-1}}(u_{i-1})$ to $\pi_{x_i}(u_i)$

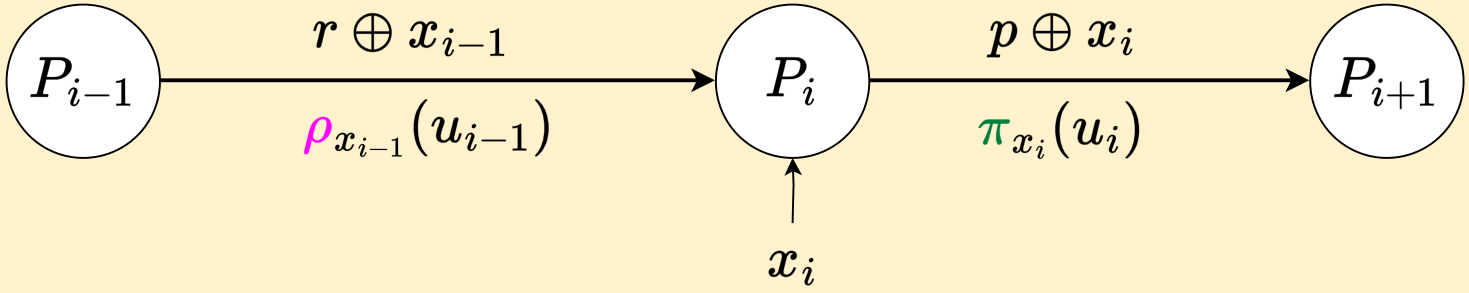


Secure evaluation - Attempt 2

- Workaround - Use two different sets of masks - π_0, π_1 instead of π
  ρ_0, ρ_1 instead of ρ

P_i needs to map $\rho_{x_{i-1}}(u_{i-1})$ to $\pi_{x_i}(u_i)$

As part of the preprocessing, P_i will be given 4 maps indexed by $(r \oplus x_{i-1}, x_i)$

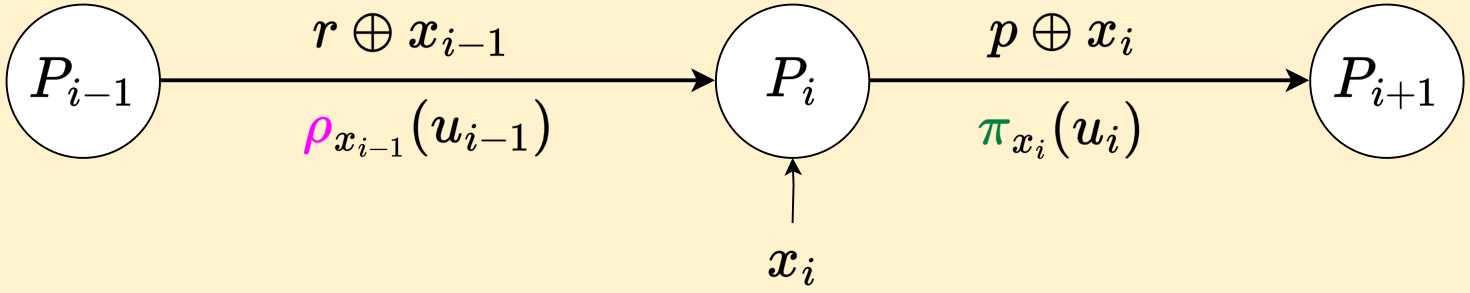


Secure evaluation - Attempt 2

- Workaround - Use two different sets of masks - π_0, π_1 instead of π
  ρ_0, ρ_1 instead of ρ

P_i needs to map $\rho_{x_{i-1}}(u_{i-1})$ to $\pi_{x_i}(u_i)$

As part of the preprocessing, P_i will be given 4 maps indexed by $(r \oplus x_{i-1}, x_i)$



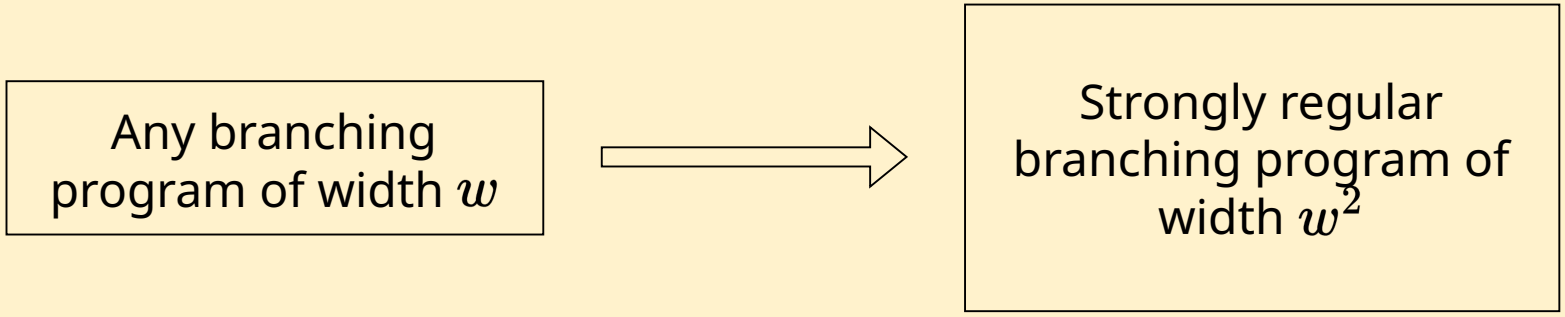
Secure for a class of BP called Strongly Regular Branching Programs!

Strongly regular branching program

- Our PSS protocol is secure for all SRBPs
- Examples: AND, XOR, Inner product

Strongly regular branching program

- Our PSS protocol is secure for all SRBPs
- Examples: AND, XOR, Inner product



Randomness cost of this protocol

Read-once SRBP

Randomness required: $\pi_0, \rho_0, \pi_1, \rho_1, r, p$

$$r, p \leftarrow \{0, 1\}$$

$$\pi_0, \rho_0, \pi_1, \rho_1 \leftarrow Permutation[w]$$

Randomness cost of this protocol

Read-once SRBP

Randomness required: $\pi_0, \rho_0, \pi_1, \rho_1, r, p$

$$r, p \leftarrow \{0, 1\}$$

$$\pi_0, \rho_0, \pi_1, \rho_1 \leftarrow Permutation[w]$$

$$O(w \log w)$$

Constant since w is constant

Randomness cost of this protocol

Read-once SRBP

Randomness required: $\pi_0, \rho_0, \pi_1, \rho_1, r, p$

$$r, p \leftarrow \{0, 1\}$$

$$\pi_0, \rho_0, \pi_1, \rho_1 \leftarrow Permutation[w]$$

$$O(w \log w)$$

Constant since w is constant

Read-k SRBP

$O(kw \log w)$ for read-k

Randomness cost of this protocol

Read-once SRBP

Randomness required: $\pi_0, \rho_0, \pi_1, \rho_1, r, p$

$$r, p \leftarrow \{0, 1\}$$

$$\pi_0, \rho_0, \pi_1, \rho_1 \leftarrow Permutation[w]$$

$$O(w \log w)$$

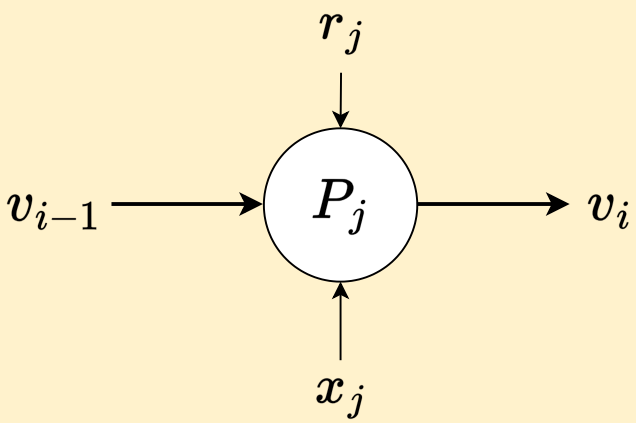
Constant since w is constant

Read-k SRBP

$O(kw \log w)$ for read-k

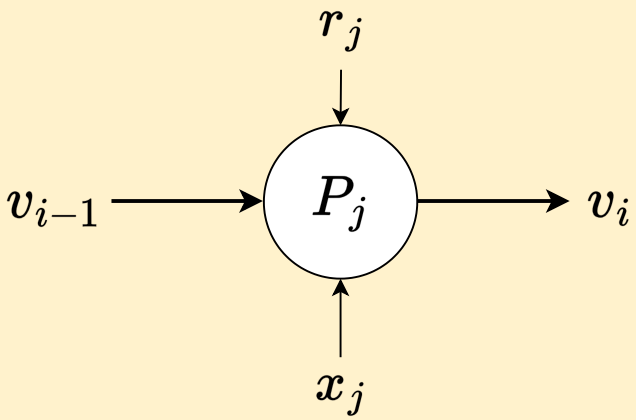
Any width-w read-k BP has a speak-$O(k)$ PSS protocol with $O(kw^2 \log w)$ randomness complexity

PSS protocols to branching programs



$$v_i = \mathsf{Nxt}(i, v_{i-1}, r_j, x_j)$$

PSS protocols to branching programs



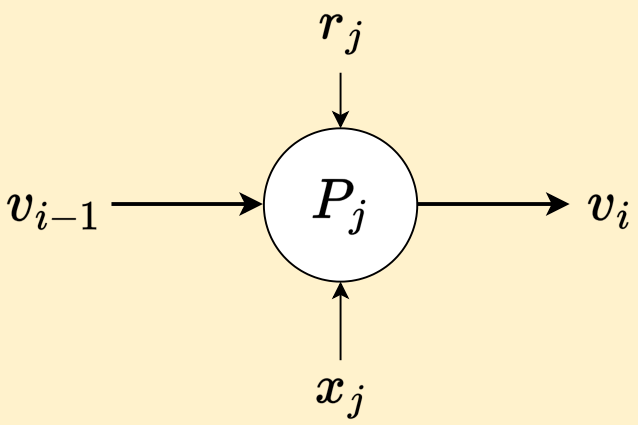
Step 1:

$$v_i = \mathsf{Nxt}(i, v_{i-1}, r_j, x_j)$$

$$\mathtt{Tr}_0^{(t)}(v_{i-1}) := \mathsf{Nxt}(i, v_{i-1}, r_j^*, 0) \qquad v_{i-1} \in M_{i-1} \qquad v_i \in M_i$$

$$\mathtt{Tr}_1^{(t)}(v_{i-1}) := \mathsf{Nxt}(i, v_{i-1}, r_j^*, 1) \qquad \mathtt{Tr}_0^{(t)}, \mathtt{Tr}_1^{(t)} : M_{i-1} \to M_i$$

PSS protocols to branching programs



Step 1:

$$v_i = \mathsf{Nxt}(i, v_{i-1}, r_j, x_j)$$

$$\mathtt{Tr}_0^{(t)}(v_{i-1}) := \mathsf{Nxt}(i, v_{i-1}, r^*, 0) \qquad v_{i-1} \in M_{i-1} \qquad v_i \in M_i$$

$$\mathtt{Tr}_1^{(t)}(v_{i-1}) := \mathsf{Nxt}(i, v_{i-1}, r^*, 1) \qquad \mathtt{Tr}_0^{(t)}, \mathtt{Tr}_1^{(t)} : M_{i-1} \to M_i$$

Step 2:

Bound $|M_i| \leq 2^{R+2}$

message space for ith message over all inputs and randomness

Following [KOR96]

constant

Summary

New simple model for MPC: PSS

SRBP: Interesting subclass of branching programs

Implications to MPC in other models:

- Simpler protocol for computing AND without preprocessing.
- For odd number of parties, randomness cost of AND matches state-of-the-art [CR22]

[CR22] Geoffroy Couteau and Adi Rosén. Random sources in private computation, ASIACRYPT 2022,