Random Sources in Secure Computation

Geoffroy Couteau, Adi Rosén

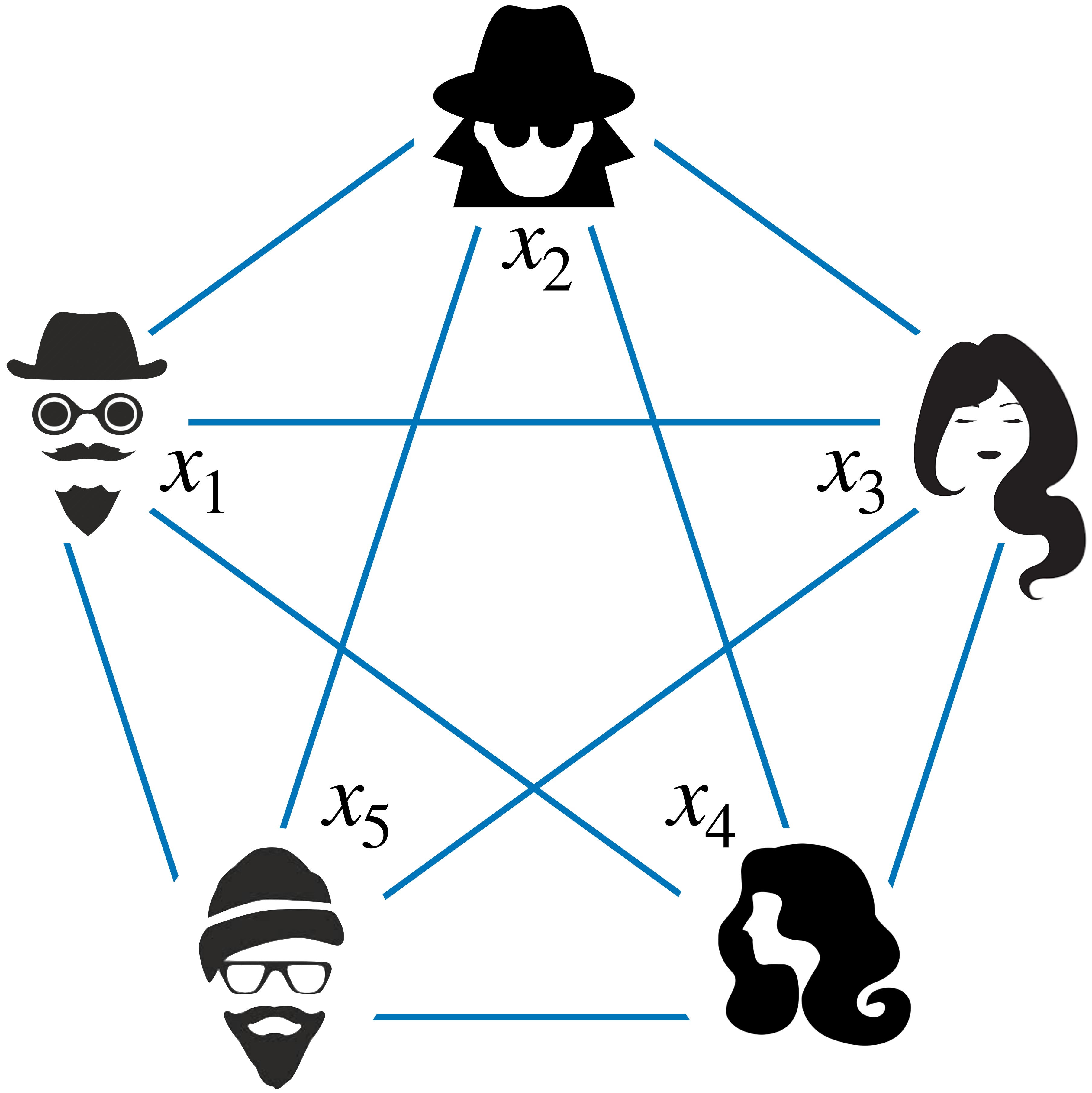
The Model

Information-Theoretic Secure Computation

- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

This Work



The Model

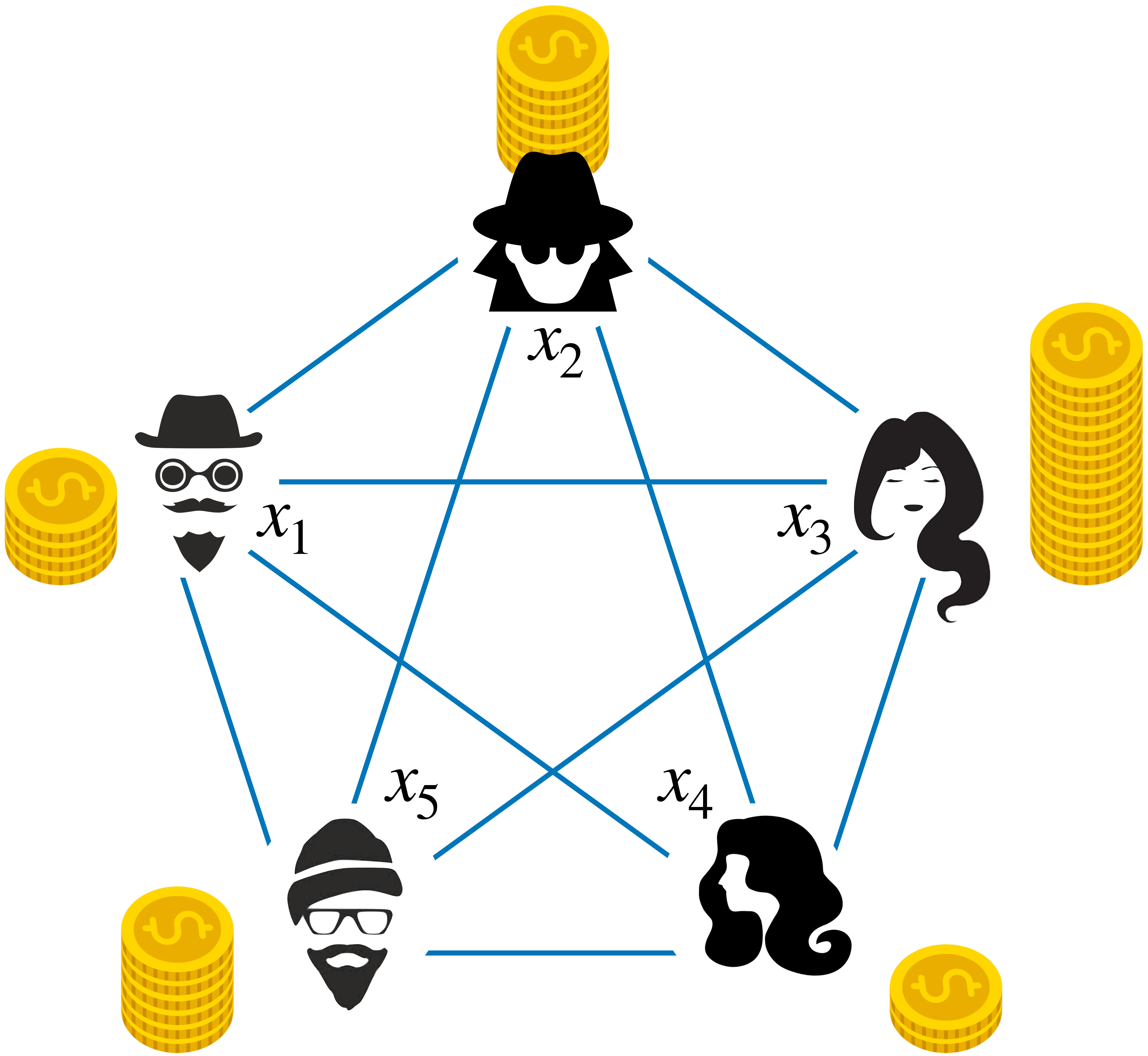
Information-Theoretic Secure Computation

- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

- Secure computation is impossible deterministically: *randomness is required*

This Work



The Model

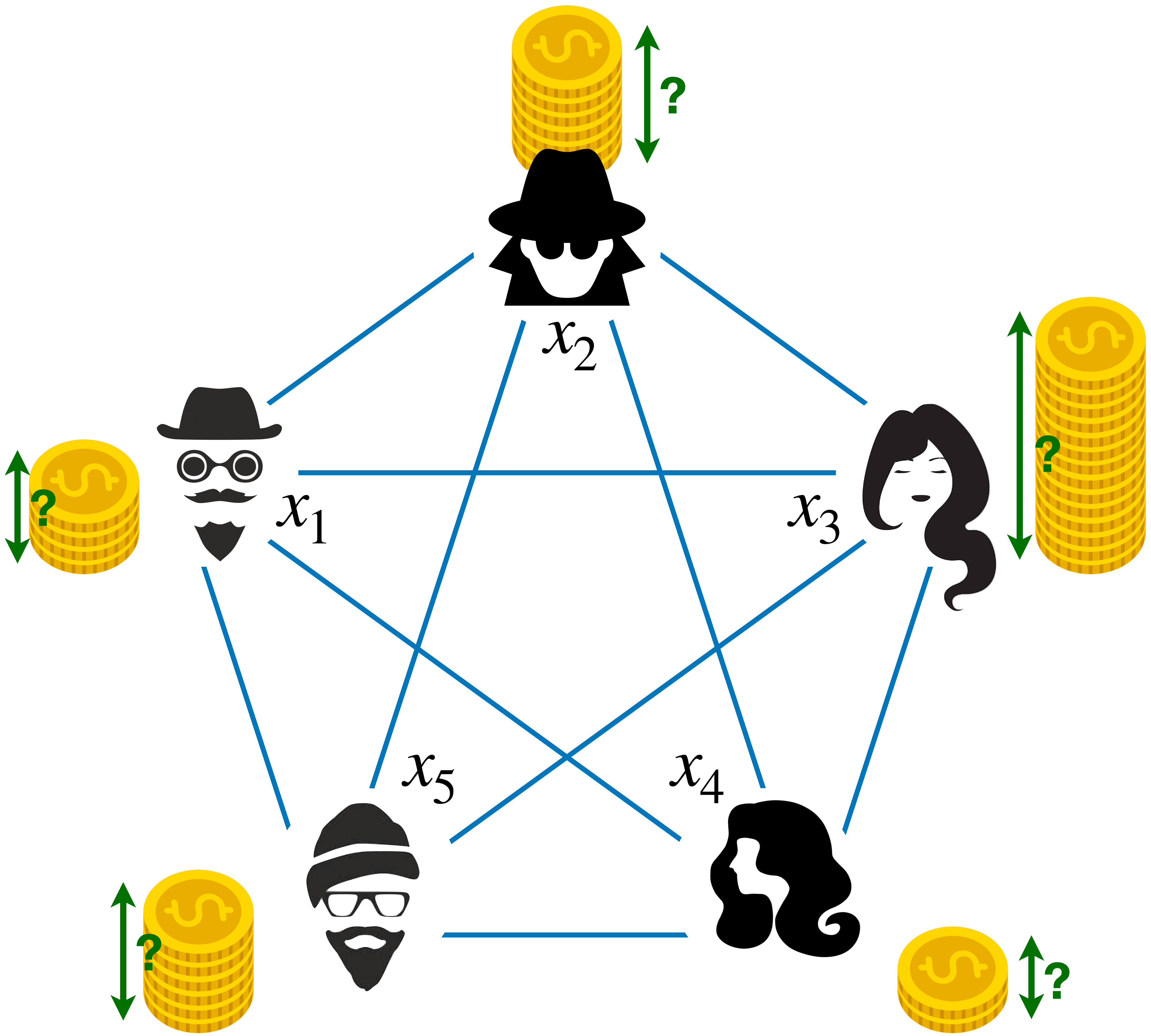
Information-Theoretic Secure Computation

- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

- Secure computation is impossible deterministically: *randomness is required*
- **Natural question:** how much randomness is needed? Studied in many previous works.

This Work



How much [coin] to compute $f(x_1, \cdots, x_n)$?

The Model

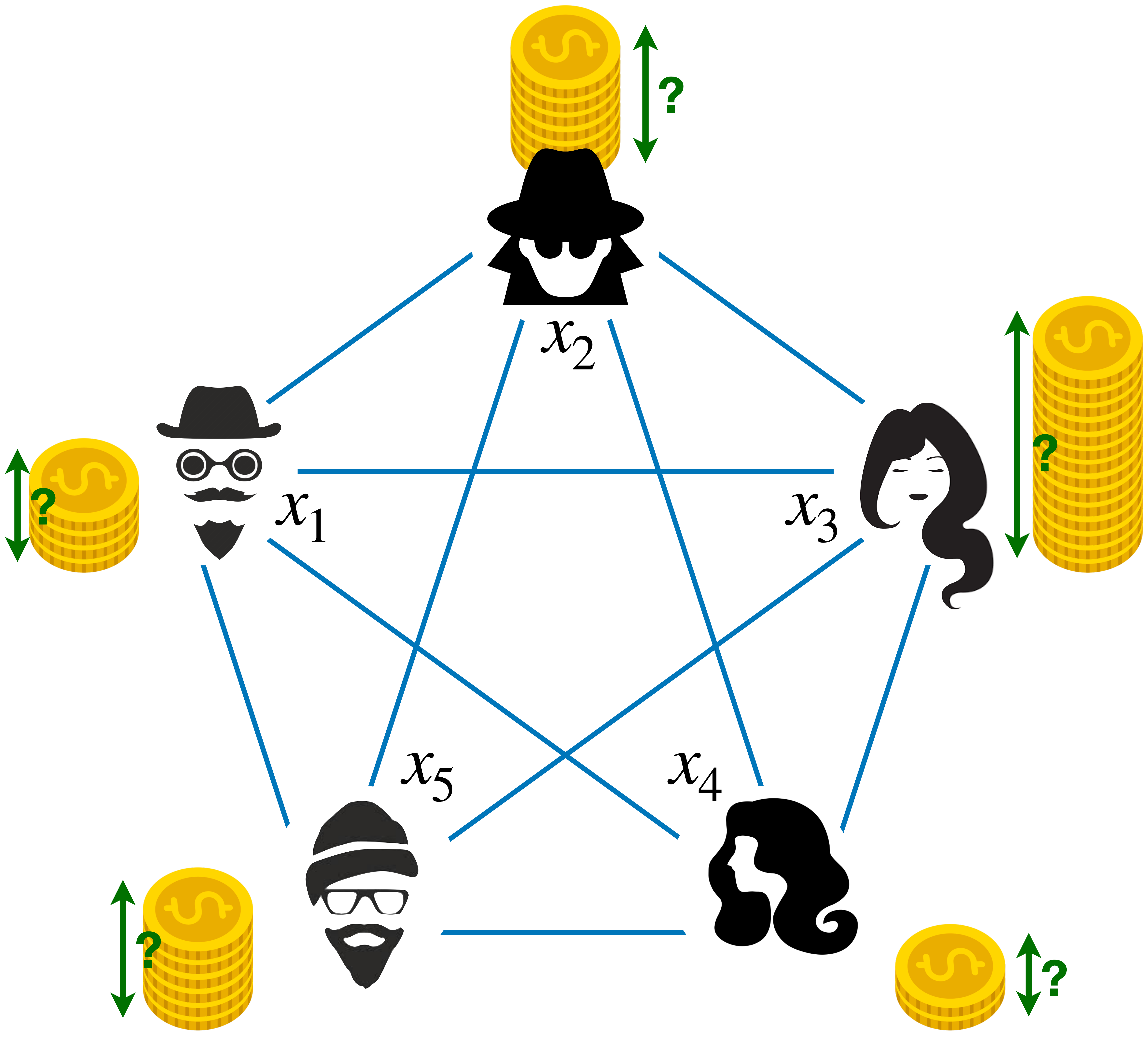
Information-Theoretic Secure Computation

- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

- Secure computation is impossible deterministically: *randomness is required*
- **Natural question:** how much randomness is needed? Studied in many previous works.
- **Motivation:** producing high-quality randomness is hard; it should be treated as a scarce resource.

This Work



How much  to compute $f(x_1, \cdots, x_n)$?

The Model

Information-Theoretic Secure Computation

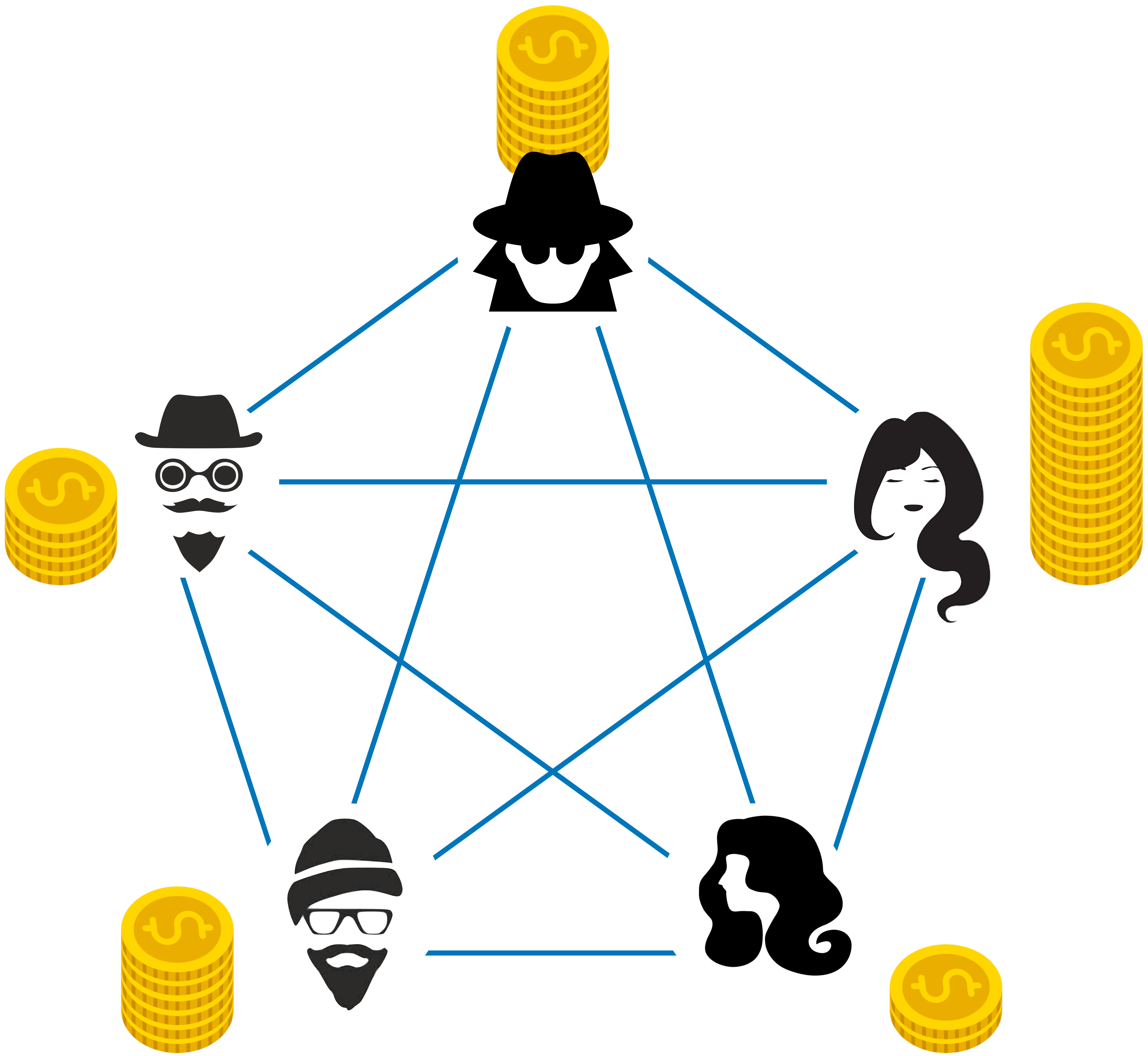
- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

- Secure computation is impossible deterministically: *randomness is required*
- **Natural question:** how much randomness is needed? Studied in many previous works.
- **Motivation:** producing high-quality randomness is hard; it should be treated as a scarce resource.

This Work

- **We ask:** *how many* players need to toss random coins?
- **Motivation:** you don't want to trust everyone's ability to toss high-quality random coins!



How many parties with [coin] to compute $f(x_1, \cdots, x_n)$?

The Model

Information-Theoretic Secure Computation

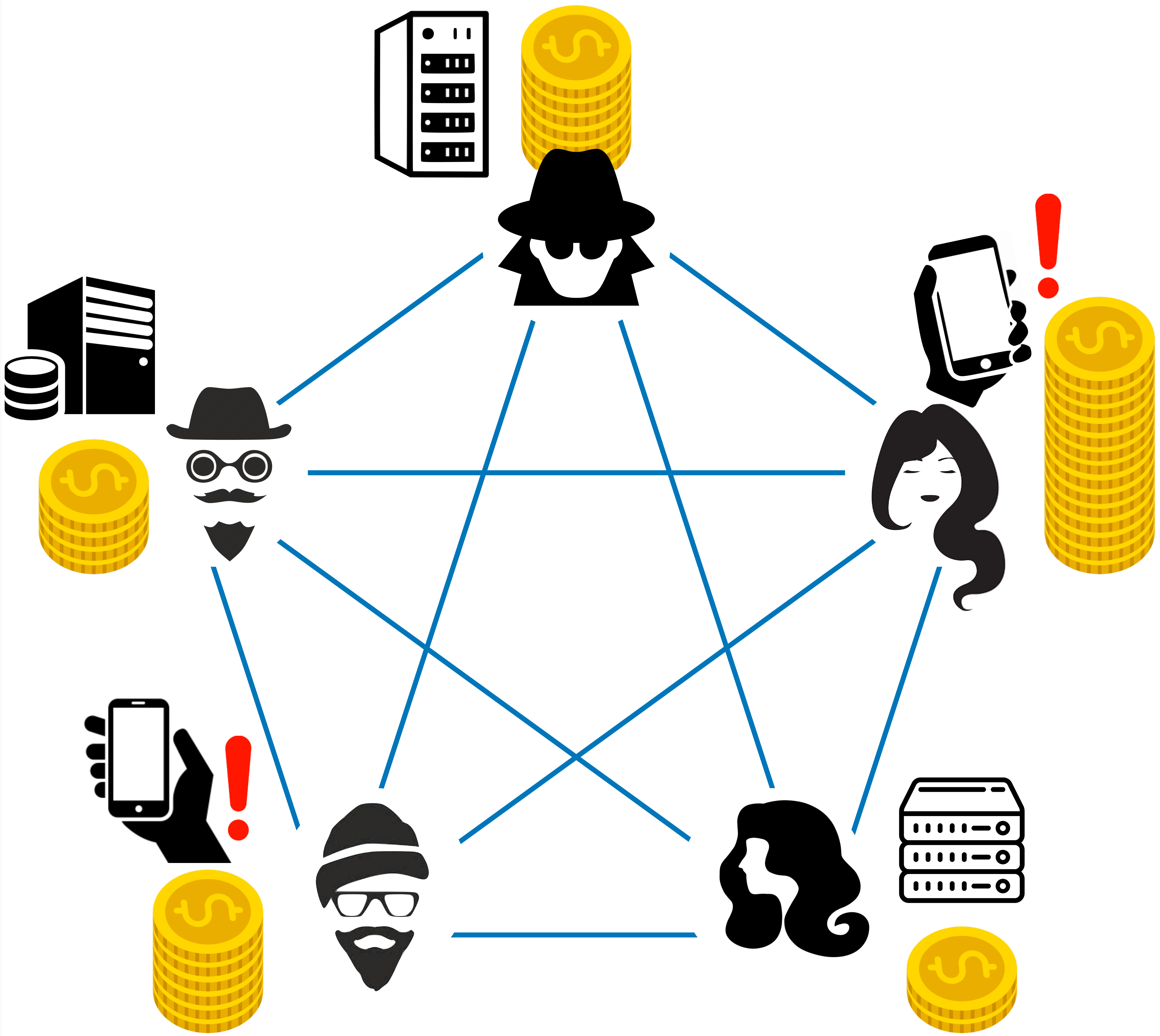
- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

- Secure computation is impossible deterministically: *randomness is required*
- **Natural question:** how much randomness is needed? Studied in many previous works.
- **Motivation:** producing high-quality randomness is hard; it should be treated as a scarce resource.

This Work

- **We ask:** *how many* players need to toss random coins?
- **Motivation:** you don't want to trust everyone's ability to toss high-quality random coins!



How many parties with  to compute $f(x_1, \cdots, x_n)$?

The Model

Information-Theoretic Secure Computation

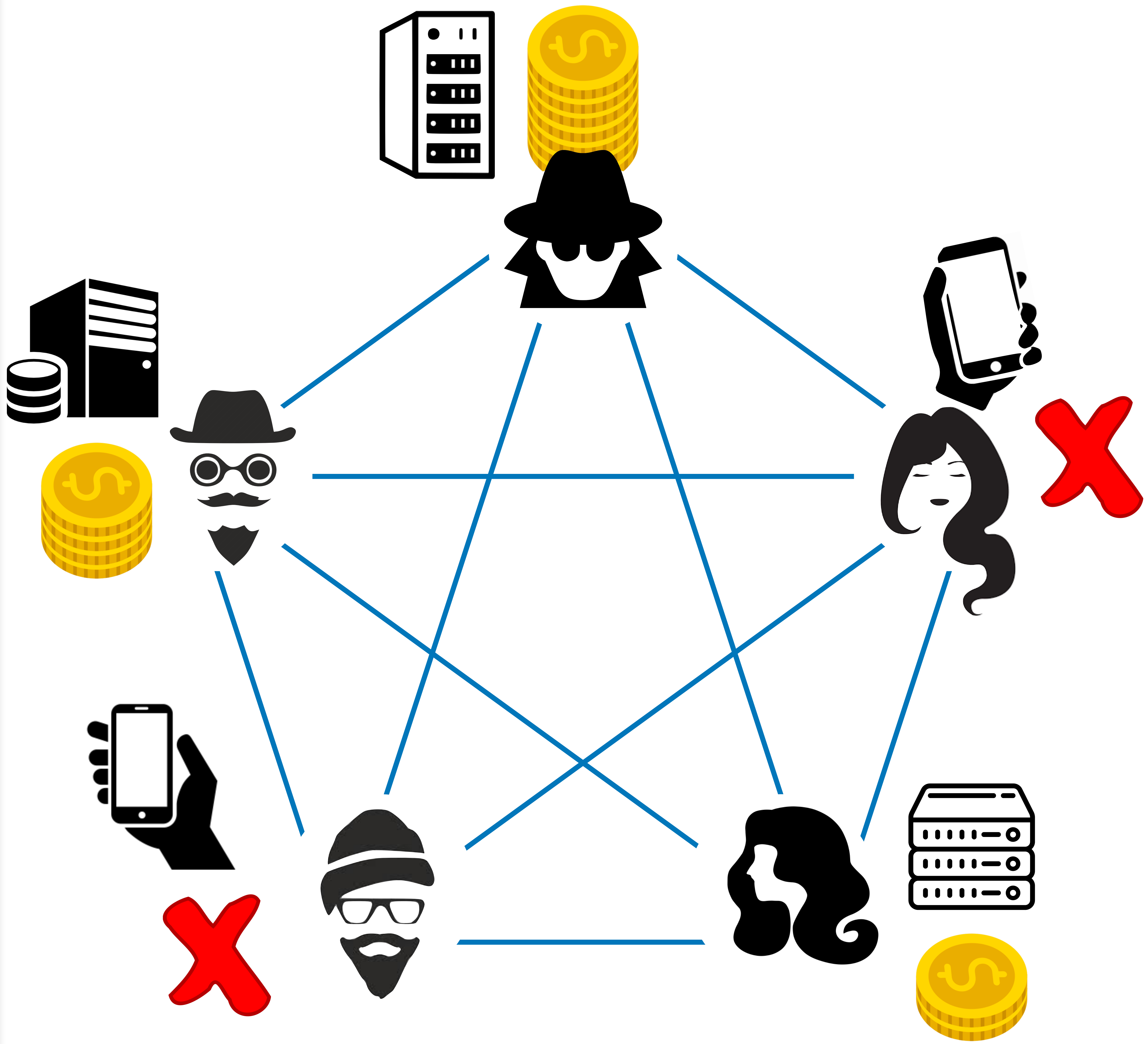
- n parties with inputs $(x_1, \cdots, x_n)$
- The adversary corrupts at most t parties
- **Goal:** computing $f(x_1, \cdots, x_n)$ without revealing more

Background

- Secure computation is impossible deterministically: *randomness is required*
- **Natural question:** how much randomness is needed? Studied in many previous works.
- **Motivation:** producing high-quality randomness is hard; it should be treated as a scarce resource.

This Work

- **We ask:** *how many* players need to toss random coins?
- **Motivation:** you don't want to trust everyone's ability to toss high-quality random coins!



How many parties with [coin] to compute $f(x_1, \cdots, x_n)$?

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

First Result: upper and lower bounds on the number of sources

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t-privately comp. with t sources

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

This work: $t+1$ sources suffice to t-privately compute any randomized functionality

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

First Result: upper and lower bounds on the number of sources

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t-privately comp. with t sources

Previous work

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

This work: $t+1$ sources suffice to t-privately compute any randomized functionality

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

First Result: upper and lower bounds on the number of sources

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t -privately comp. with t sources

Previous work

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

Easy

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

This work: $t+1$ sources suffice to t -privately compute any randomized functionality

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

First Result: upper and lower bounds on the number of sources

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t -privately comp. with t sources

Previous work

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

Easy

Core result

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

This work: $t+1$ sources suffice to t -privately compute any randomized functionality

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

First Result: upper and lower bounds on the number of sources

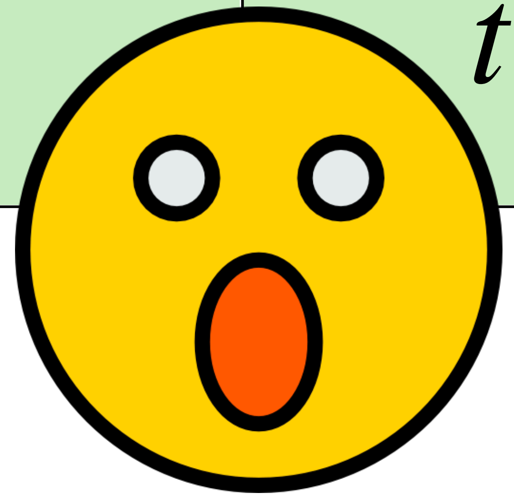
This work: there is a randomized functionality $\mathscr{F}$ that cannot be t-privately comp. with t sources

Previous work

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

Easy

Core result



This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

Secure computation is impossible deterministically… But it is possible even if we allow the adversary to corrupt *all* players that toss coins!

This work: $t+1$ sources suffice to t-privately compute any randomized functionality

Second Result: Randomness vs Random Sources, and 1-Private AND

Second Result: the randomness complexity of 1-private AND

- **Question.** What is the tradeoff between the number of sources and the randomness complexity? Do we need much more randomness to use a minimal number of sources?

  Proving tight bounds on randomness is notoriously very hard. Towards making progress, as in previous works, we focus on a natural functionality: the n-party AND.

  - **Best known protocol for 1-private, n-party AND:** 8 bits, 2 sources (KOPRTV, TCC'19)
  - **Question.** Can we match this bound with a single source?
  - **Our result.** Surprisingly, we manage to improve *both* the randomness complexity and the number of sources: we describe a protocol using only 6 bits and a single source.

First Result: Random Source Complexity of t–Private Computation

Kushilevitz & Mansour, PODC’96 show that t parties must toss coins for t-private XOR

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t-privately comp. with t sources

Previous work

Core result

Easy

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

This work: $t+1$ sources suffice to t-privately compute any randomized functionality

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

Previous work

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t -privately comp. with t sources

Easy

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

Core result

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

Starting point (warmup)

This work: $t+1$ sources suffice to t -privately compute any randomized functionality

First Result: Random Source Complexity of t-Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

Previous work

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t-privately comp. with t sources

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

Easy

Core result

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

Most of the presentation

This work: $t+1$ sources suffice to t-privately compute any randomized functionality

First Result: Random Source Complexity of t–Private Computation

Kushilevitz & Mansour, PODC'96 show that t parties must toss coins for t-private XOR

Previous work

Easy - see paper!

This work: there is a randomized functionality $\mathscr{F}$ that cannot be t-privately comp. with t sources

	Deterministic Functionalities	Randomized Functionalities
Lower Bound	t	$t+1$
Upper Bound	t	$t+1$

Easy

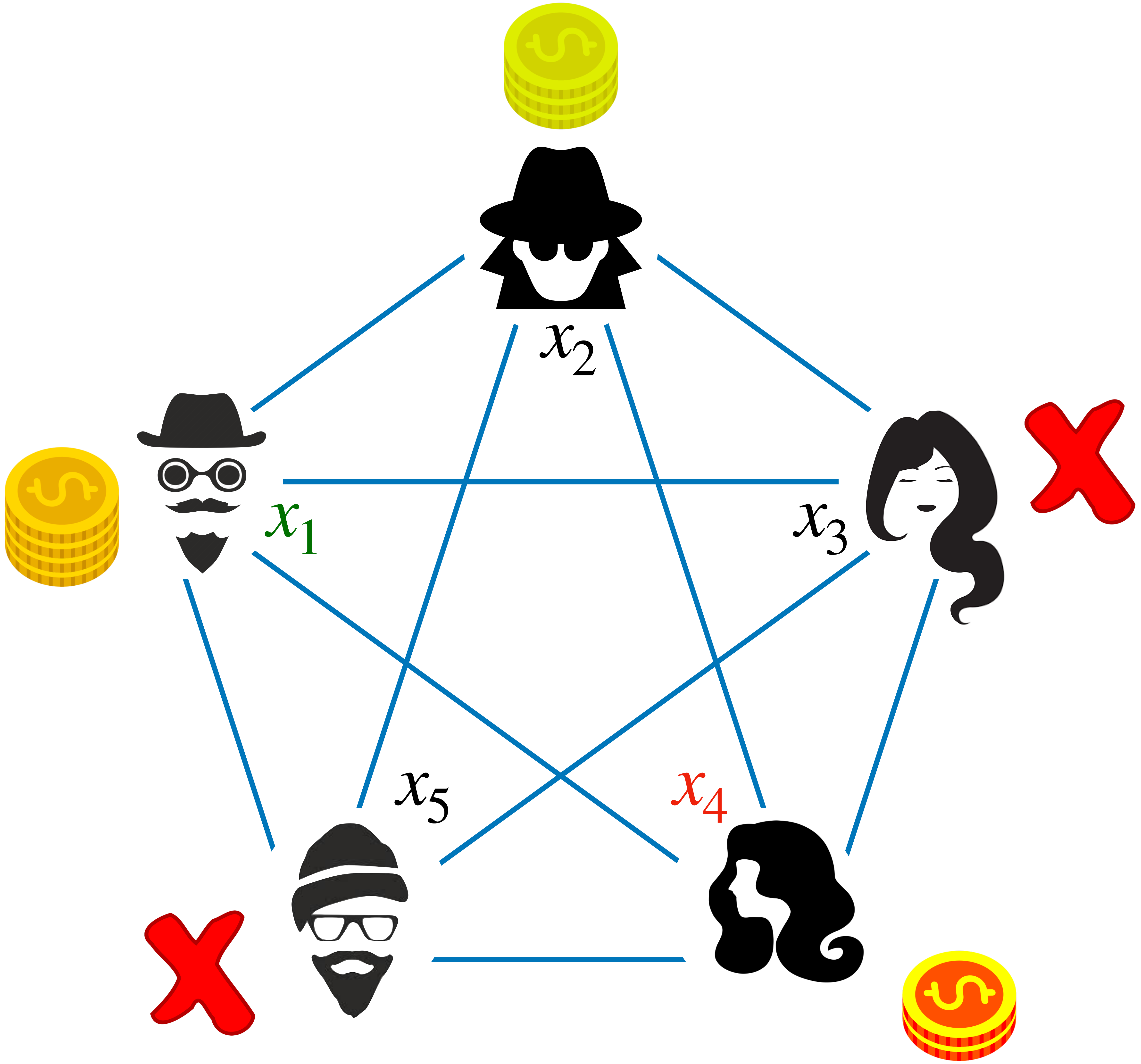
Core result

This work: It *suffices* that t parties toss coins to t-privately compute all deterministic functionalities

This work: $t+1$ sources suffice to t-privately compute any randomized functionality

Warmup: $(t+1)$ sources, randomized functionalities

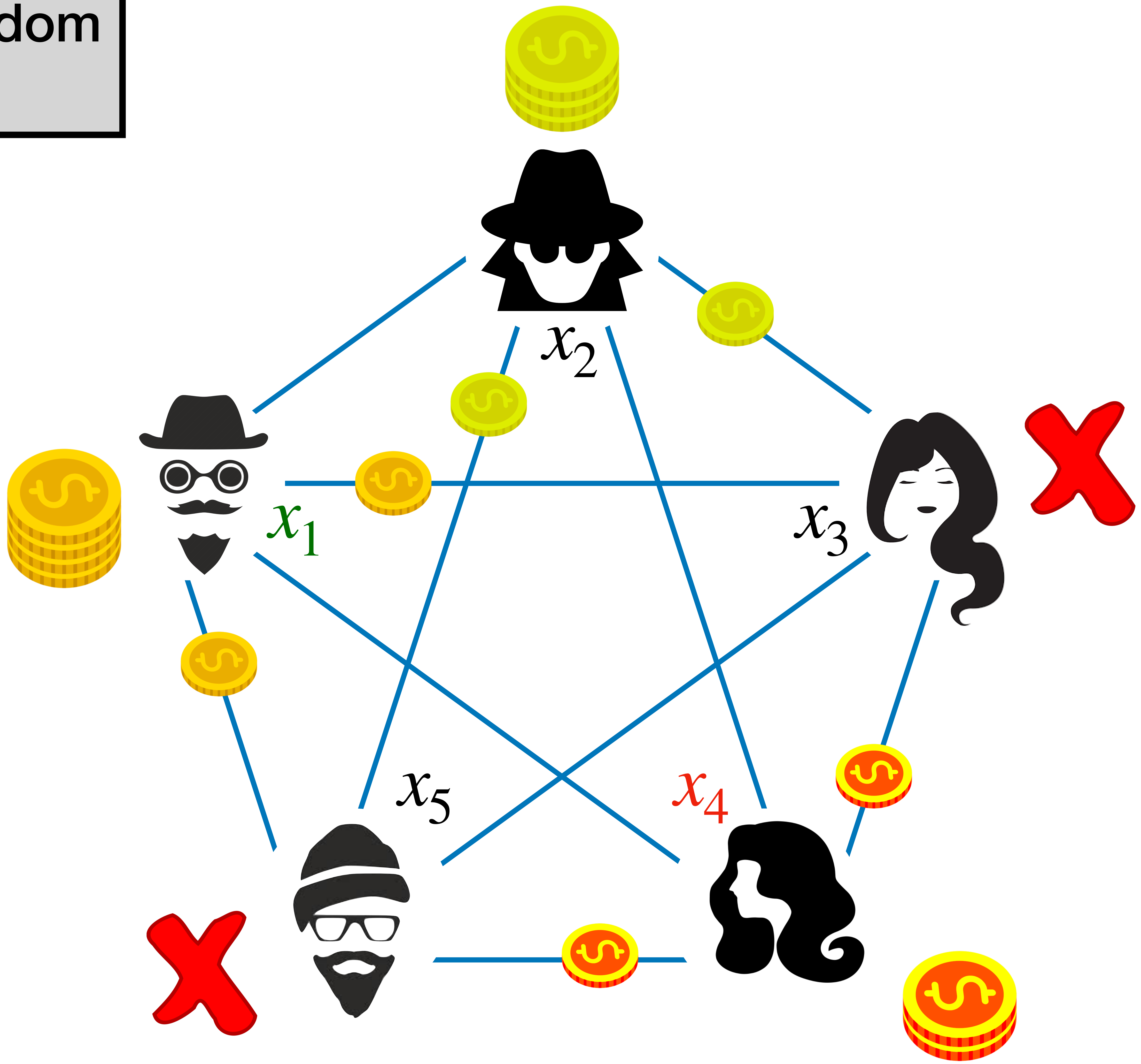
$$F(x_1, x_2, x_3, x_4, x_5; r)$$



Warmup: $(t+1)$ sources, randomized functionalities

Each source sends a random tape to each player

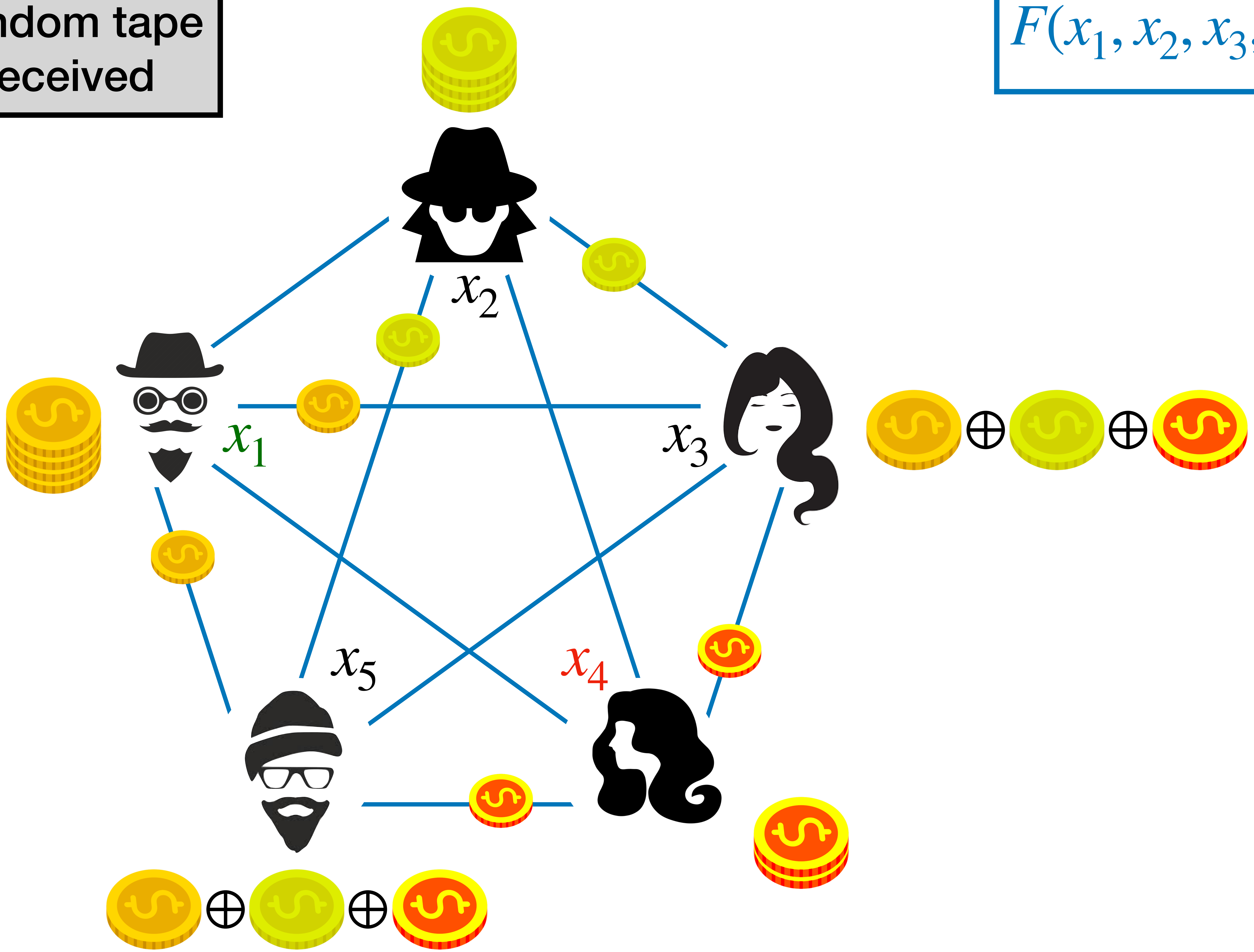
$F(x_1, x_2, x_3, x_4, x_5; r)$



Warmup: $(t+1)$ sources, randomized functionalities

Each player sets their random tape to the XOR of the tapes received

$F(x_1, x_2, x_3, x_4, x_5; r)$

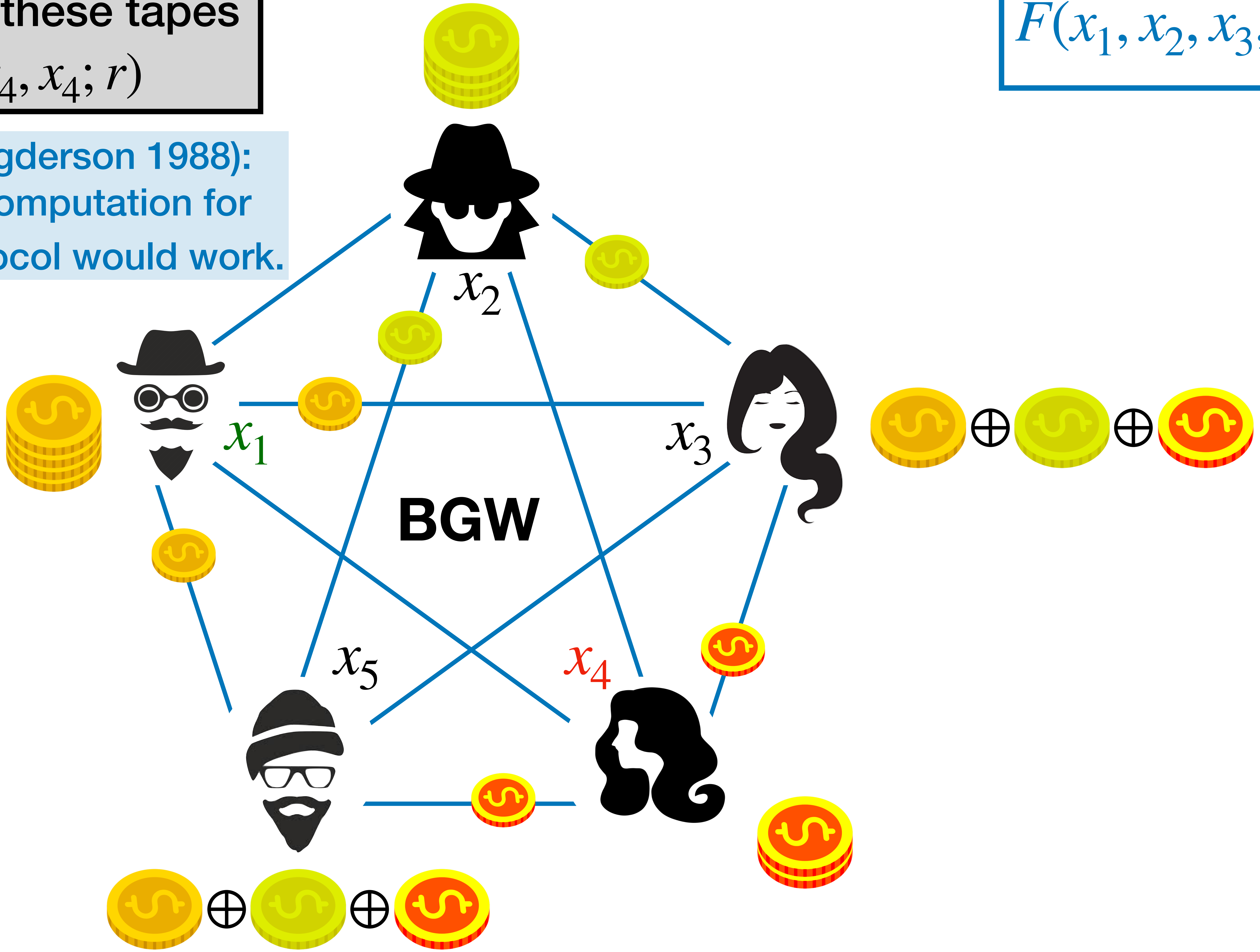


Warmup: $(t+1)$ sources, randomized functionalities

All parties run BGW with these tapes to compute $F(x_1, x_2, x_3, x_4, x_4; r)$

BGW (Ben-Or, Goldwasser, Wigderson 1988): information-theoretic secure computation for any $t < n/2$. Any other IT protocol would work.

$F(x_1, x_2, x_3, x_4, x_5; r)$

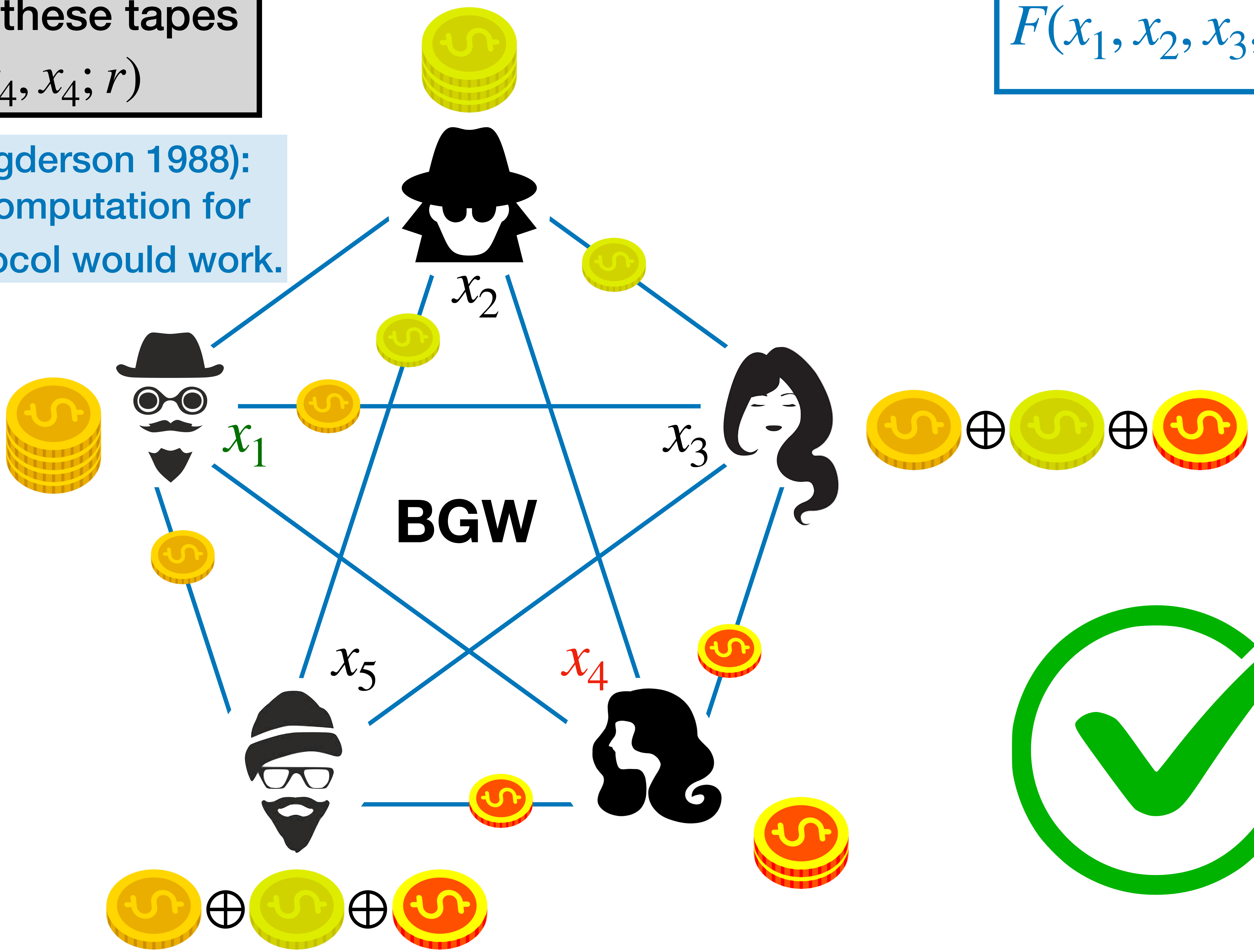


Warmup: $(t+1)$ sources, randomized functionalities

All parties run BGW with these tapes to compute $F(x_1, x_2, x_3, x_4, x_4; r)$

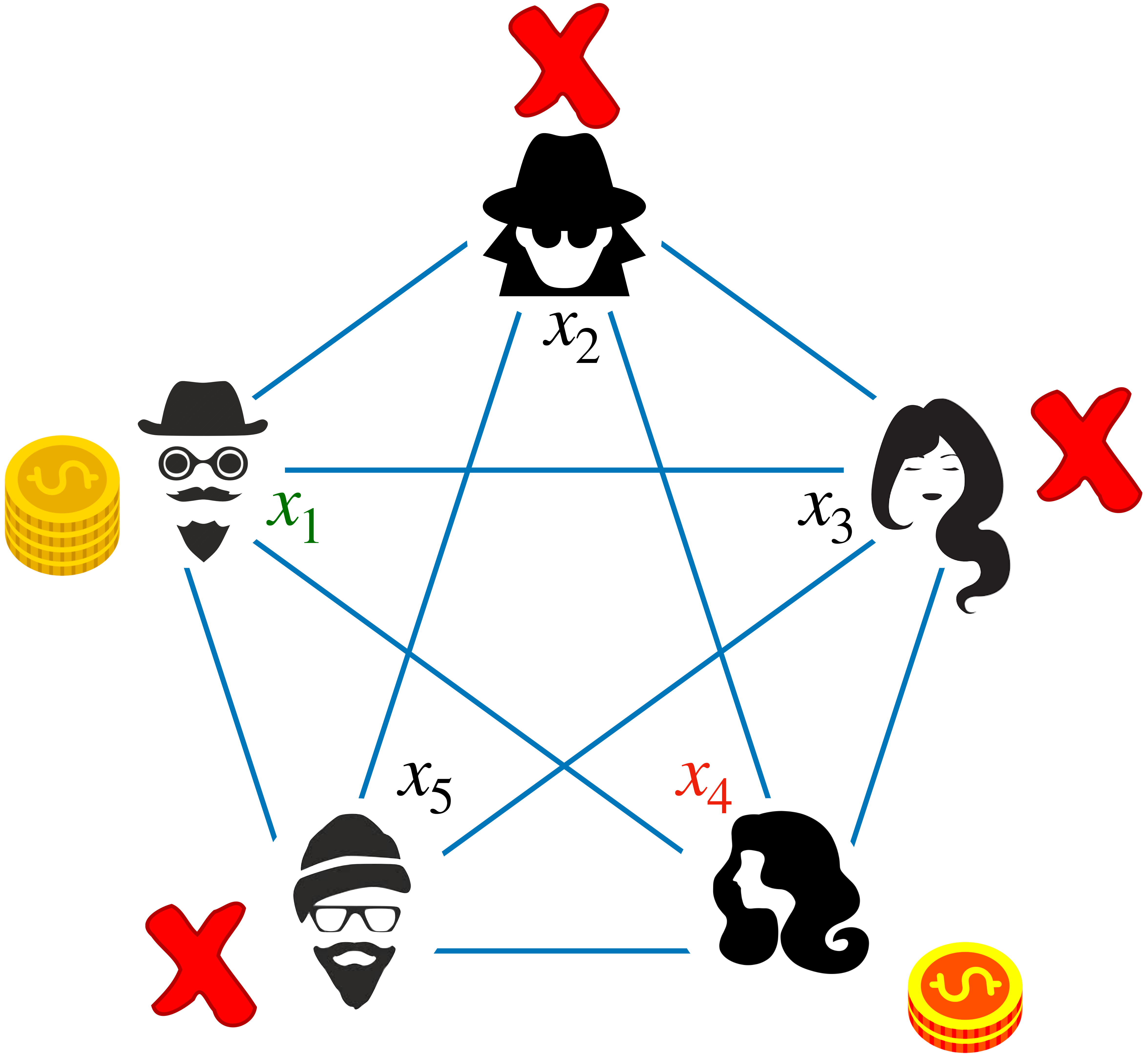
BGW (Ben-Or, Goldwasser, Wigderson 1988): information-theoretic secure computation for any $t < n/2$. Any other IT protocol would work.

$F(x_1, x_2, x_3, x_4, x_5; r)$



Core Result: t sources, deterministic functionalities

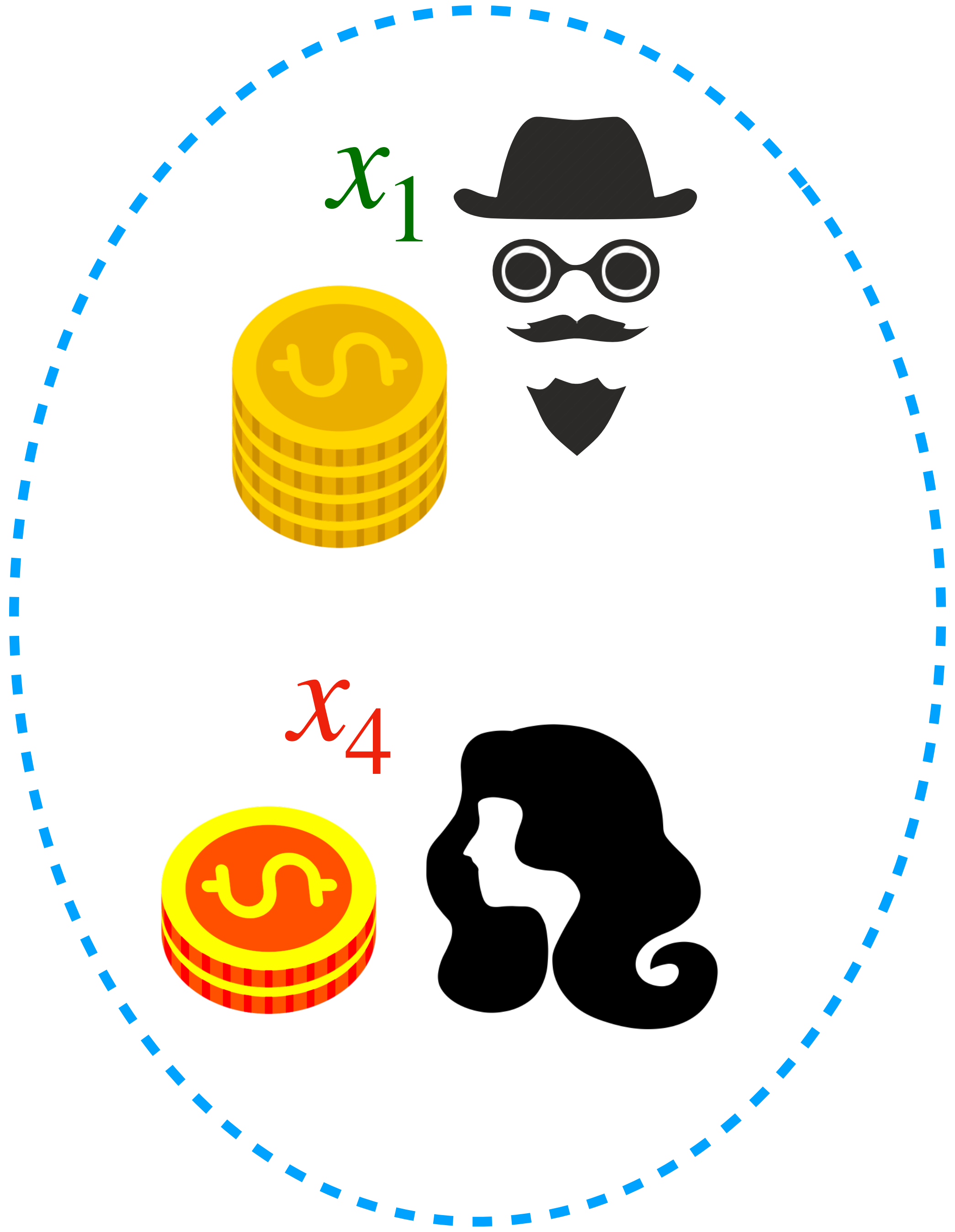
$$F(x_1, x_2, x_3, x_4, x_5)$$

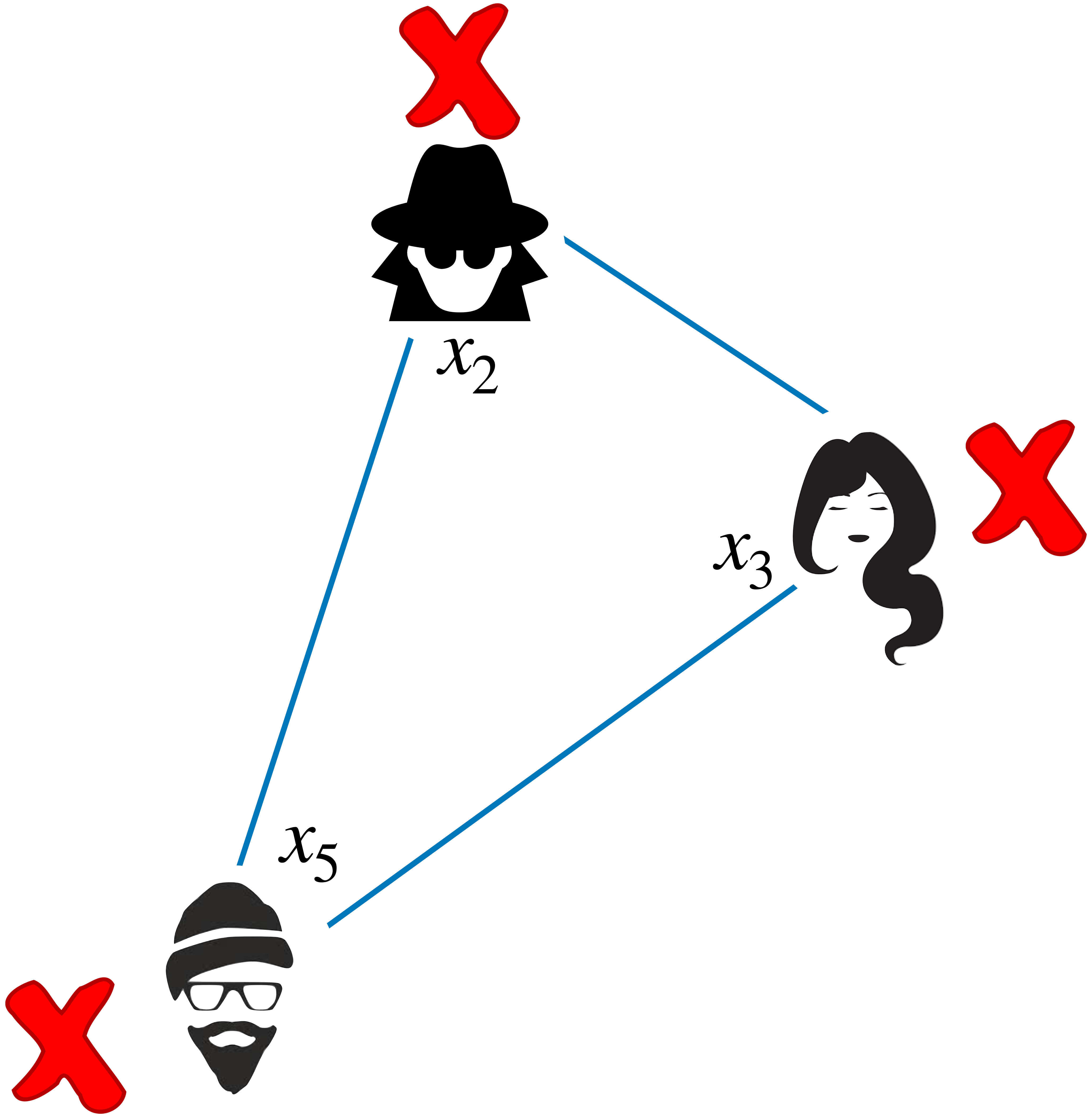


Core Result: t sources, deterministic functionalities

Isolating the sources

$F(x_1, x_2, x_3, x_4, x_5)$

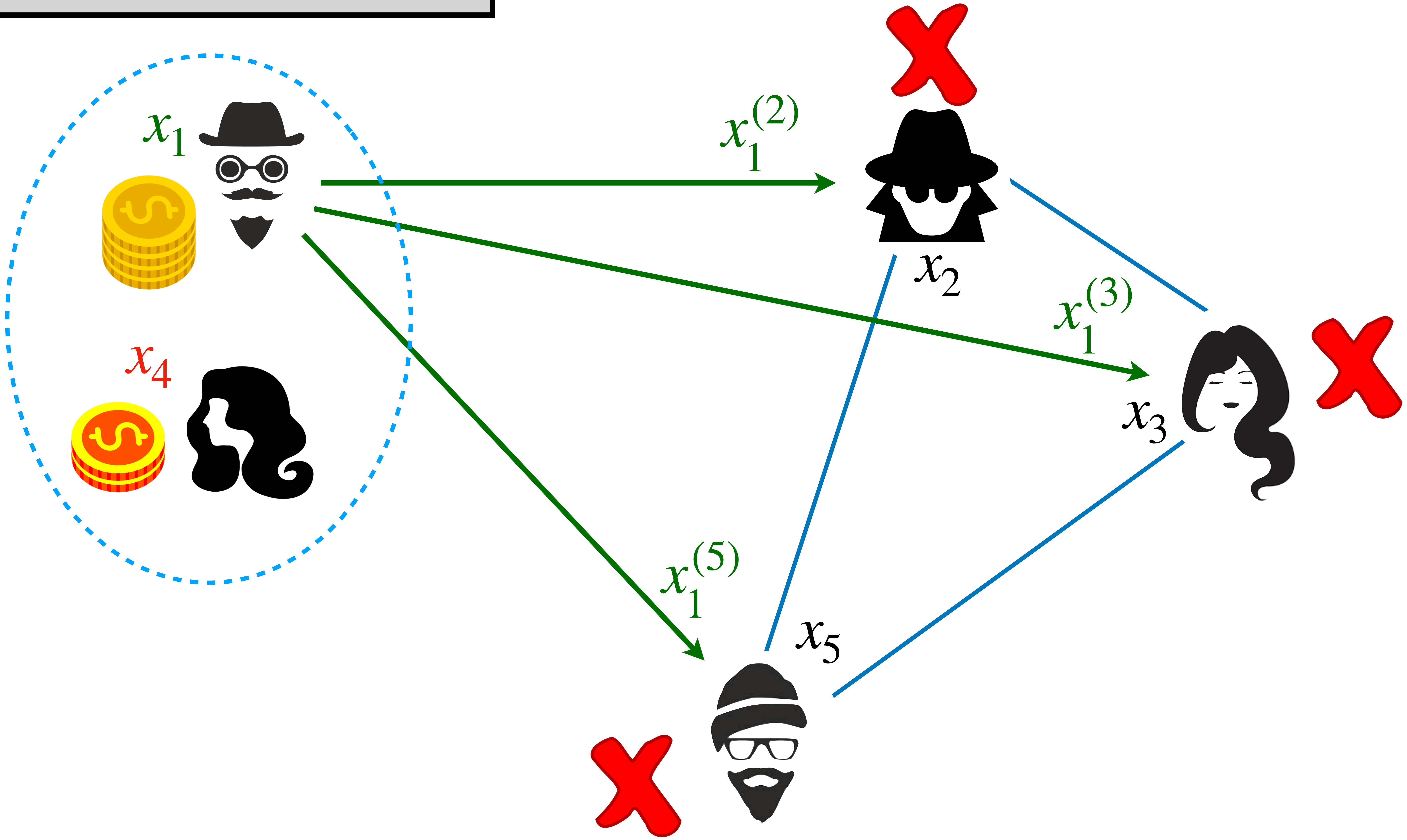




Core Result: t sources, deterministic functionalities

Input Sharing

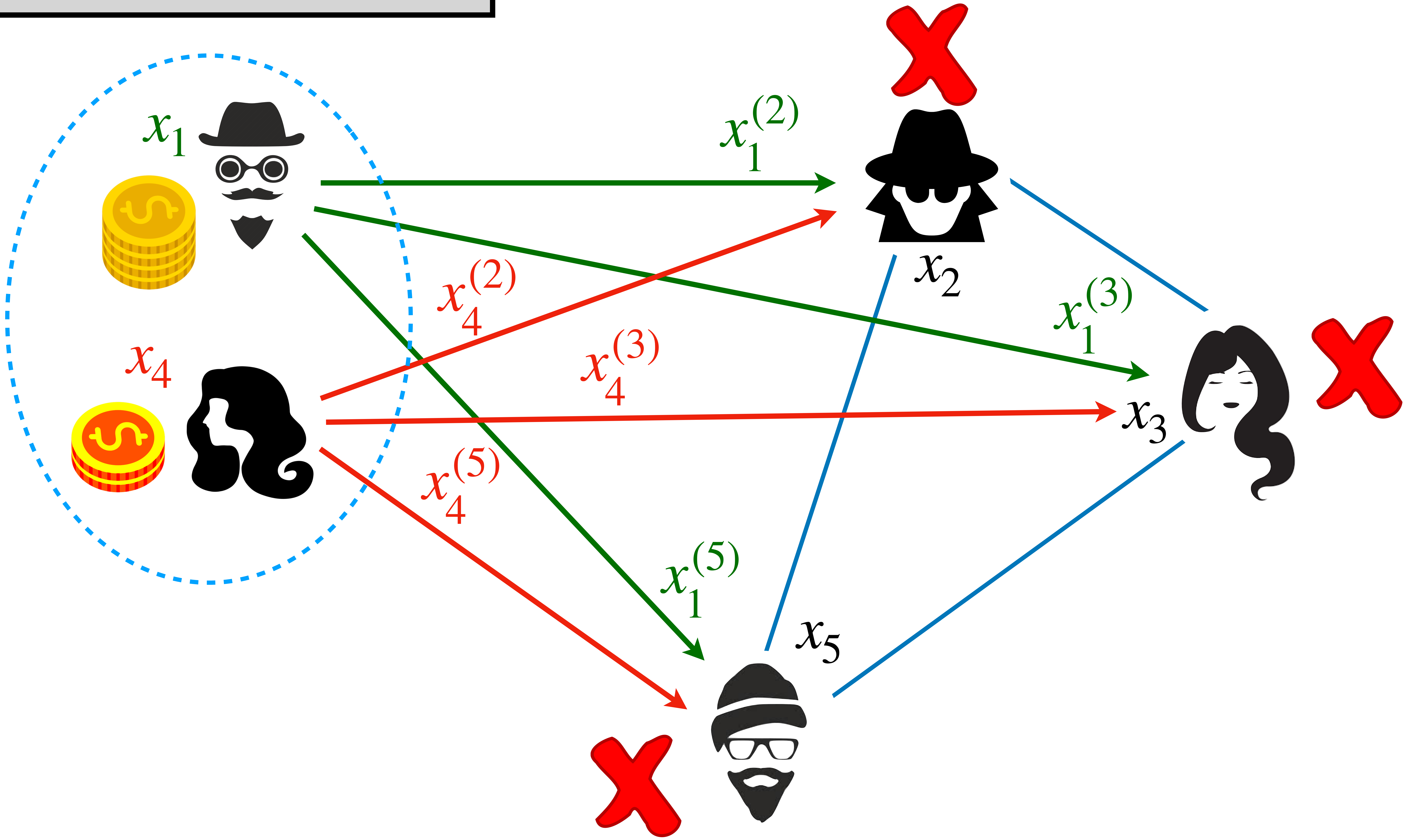
$F(x_1, x_2, x_3, x_4, x_5)$



Core Result: t sources, deterministic functionalities

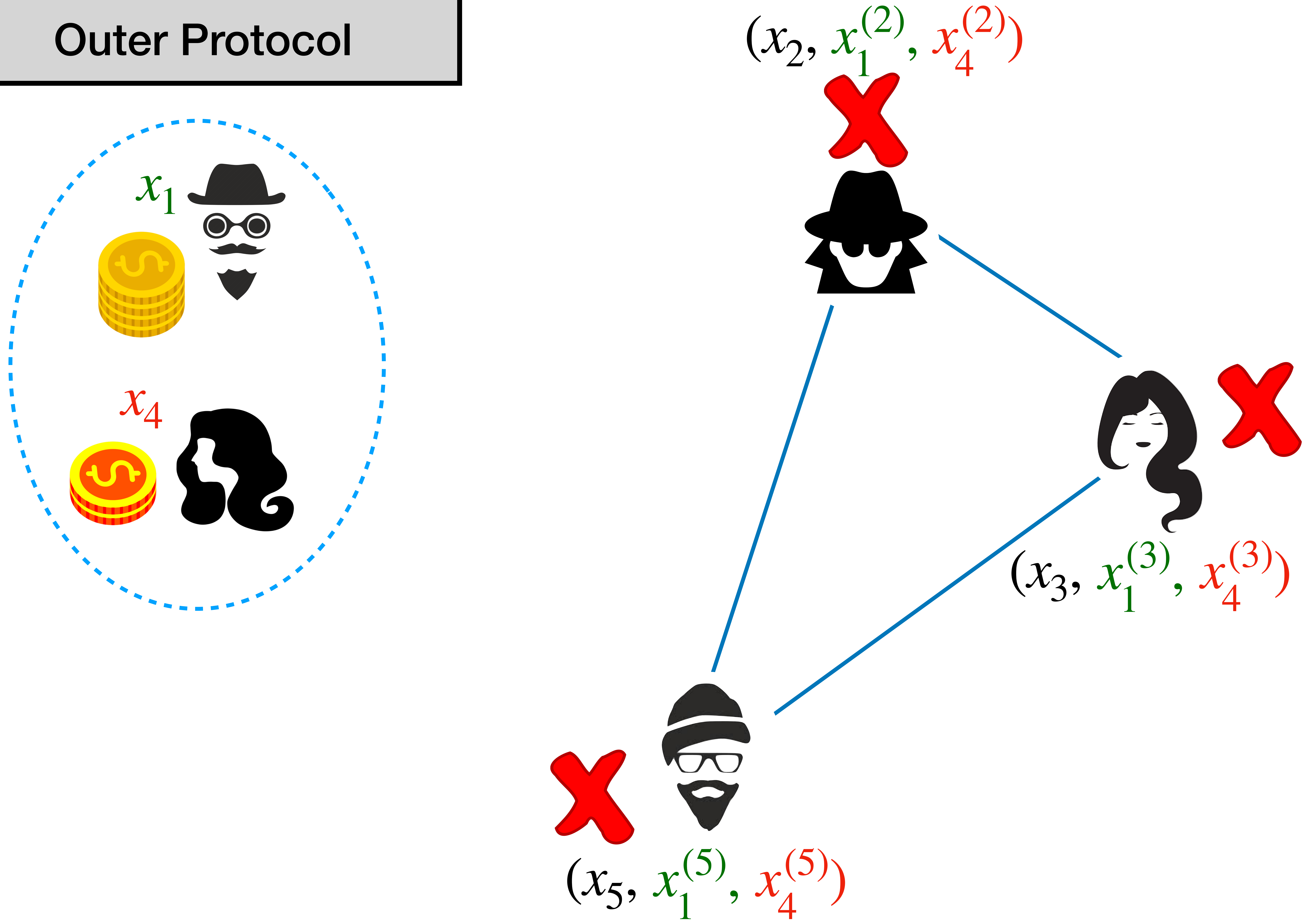
Input Sharing

$F(x_1, x_2, x_3, x_4, x_5)$



Core Result: t sources, deterministic functionalities

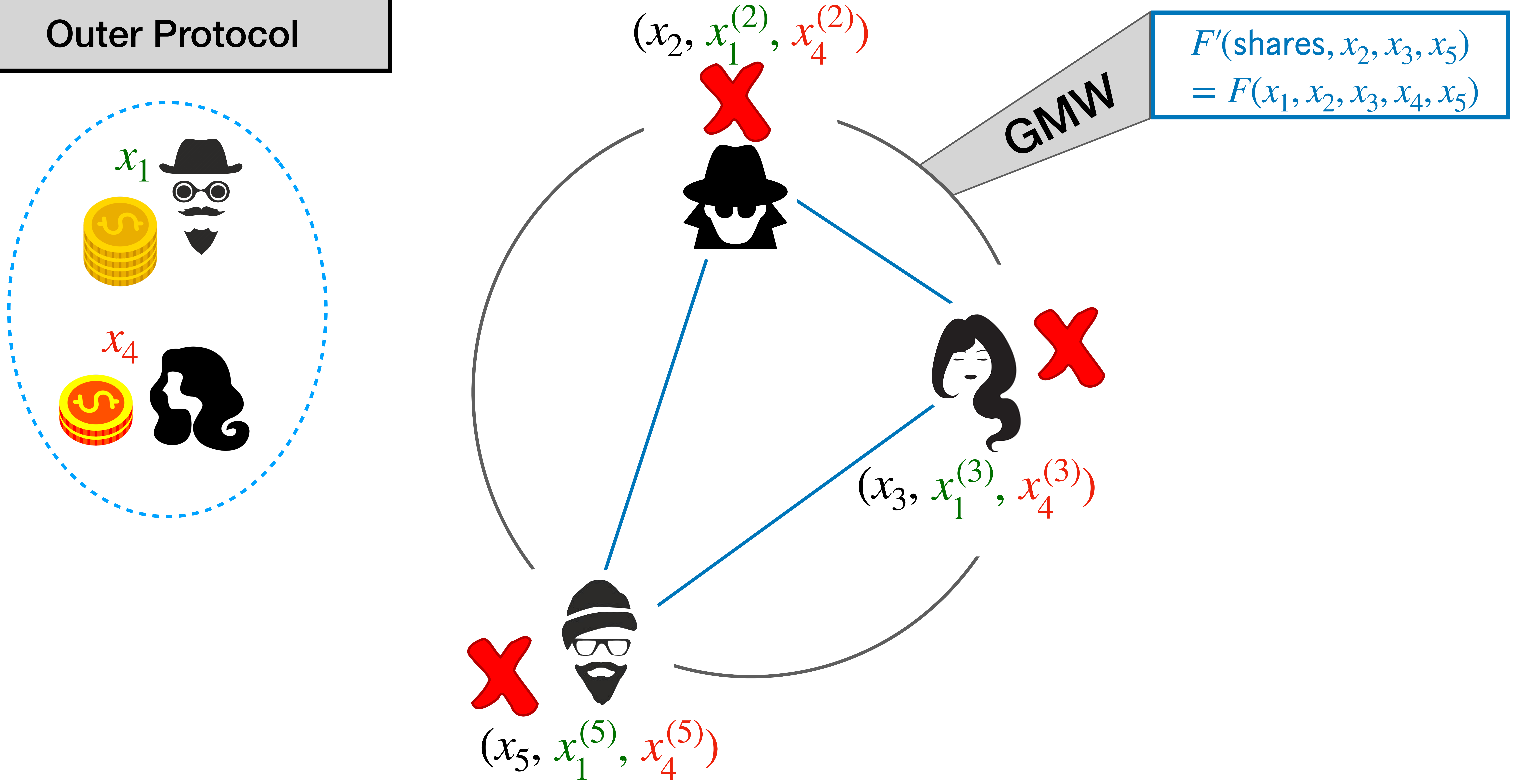
Outer Protocol



$$F'(\text{shares}, x_2, x_3, x_5)$$
$$= F(x_1, x_2, x_3, x_4, x_5)$$

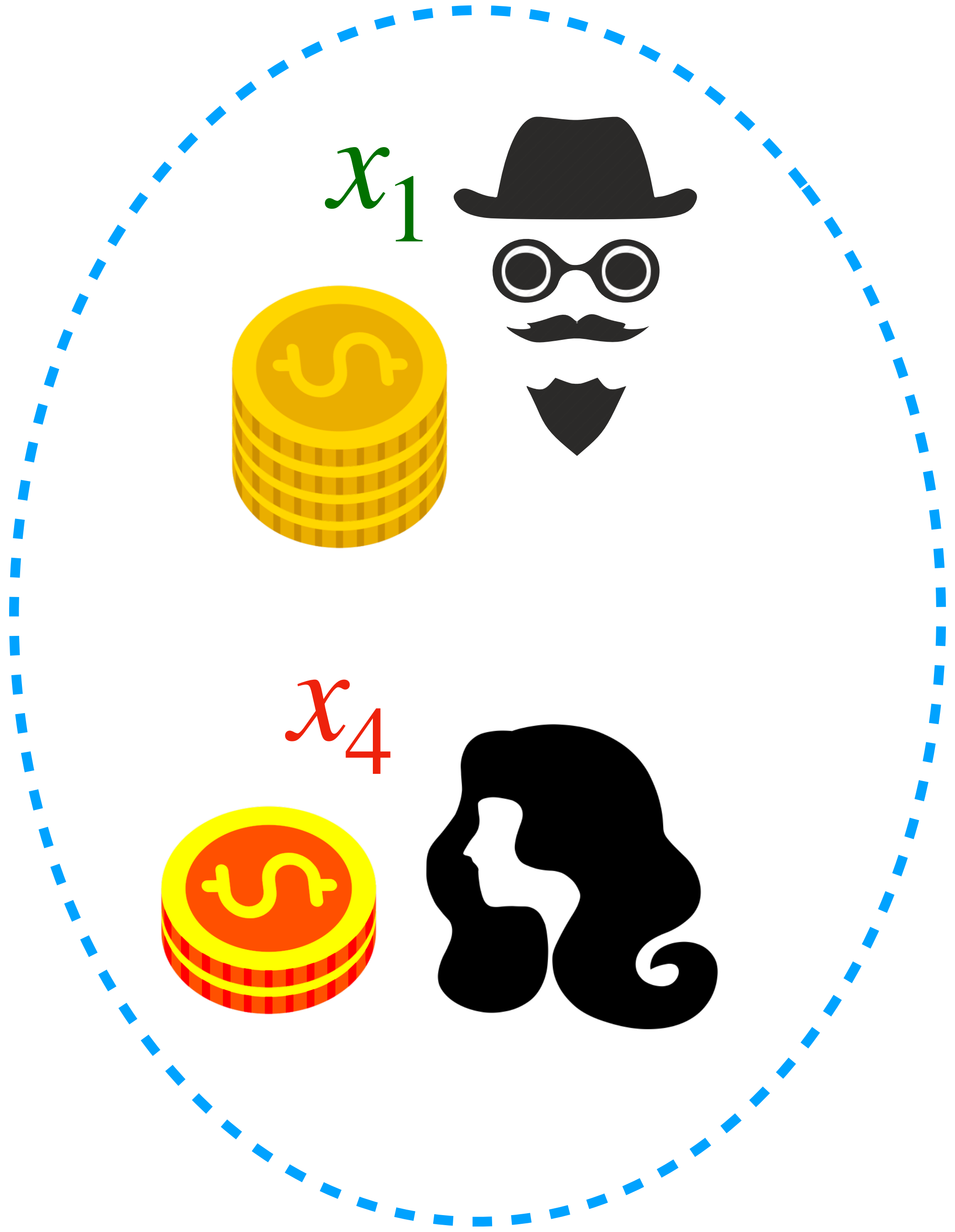
Core Result: t sources, deterministic functionalities

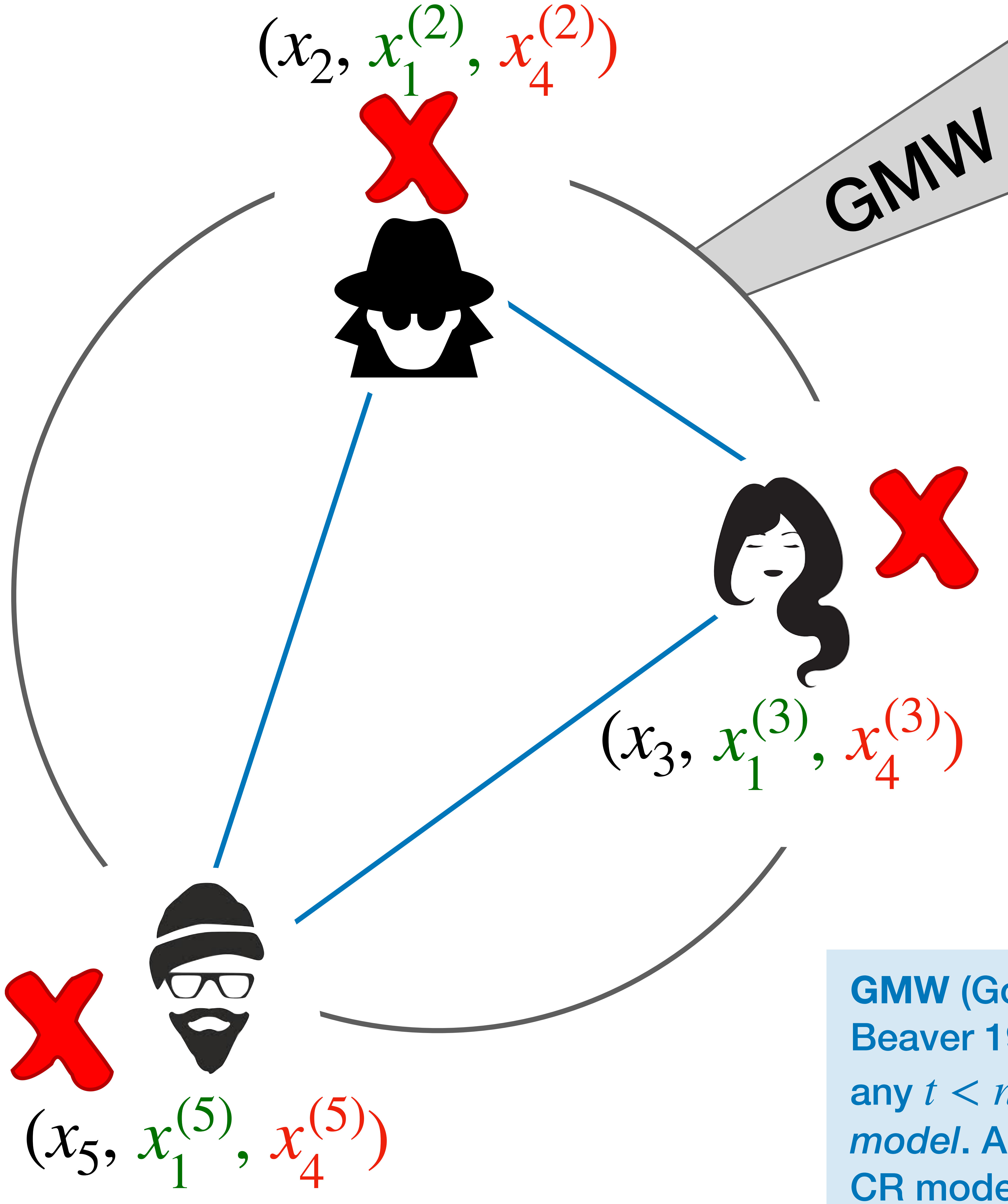
Outer Protocol



Core Result: t sources, deterministic functionalities

Outer Protocol





$F'(\text{shares}, x_2, x_3, x_5)$
$= F(x_1, x_2, x_3, x_4, x_5)$

GMW computes the circuit gate-by-gate on shares of the input wires.

XOR gates are local.

AND gates can be handled by *consuming a Beaver triple*

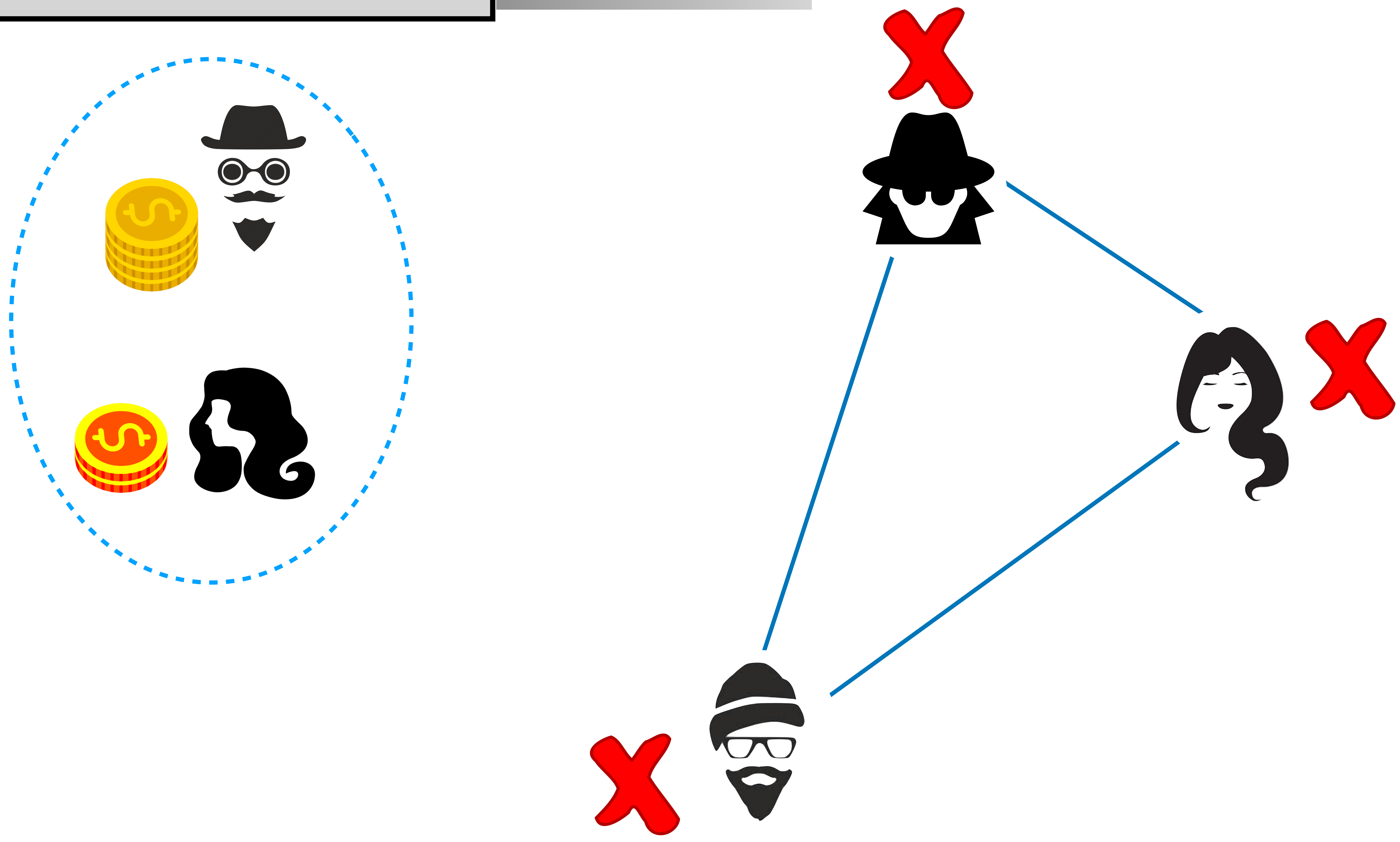
Use an *Inner Protocol*

GMW (Goldreich, Micali, Wigderson 1987 + Beaver 1995): IT secure computation for any $t < n - 1$ in the *correlated randomness model*. Any all-but-one IT protocol in the CR model would work.

Core Result: t sources, deterministic functionalities

Inner Protocol

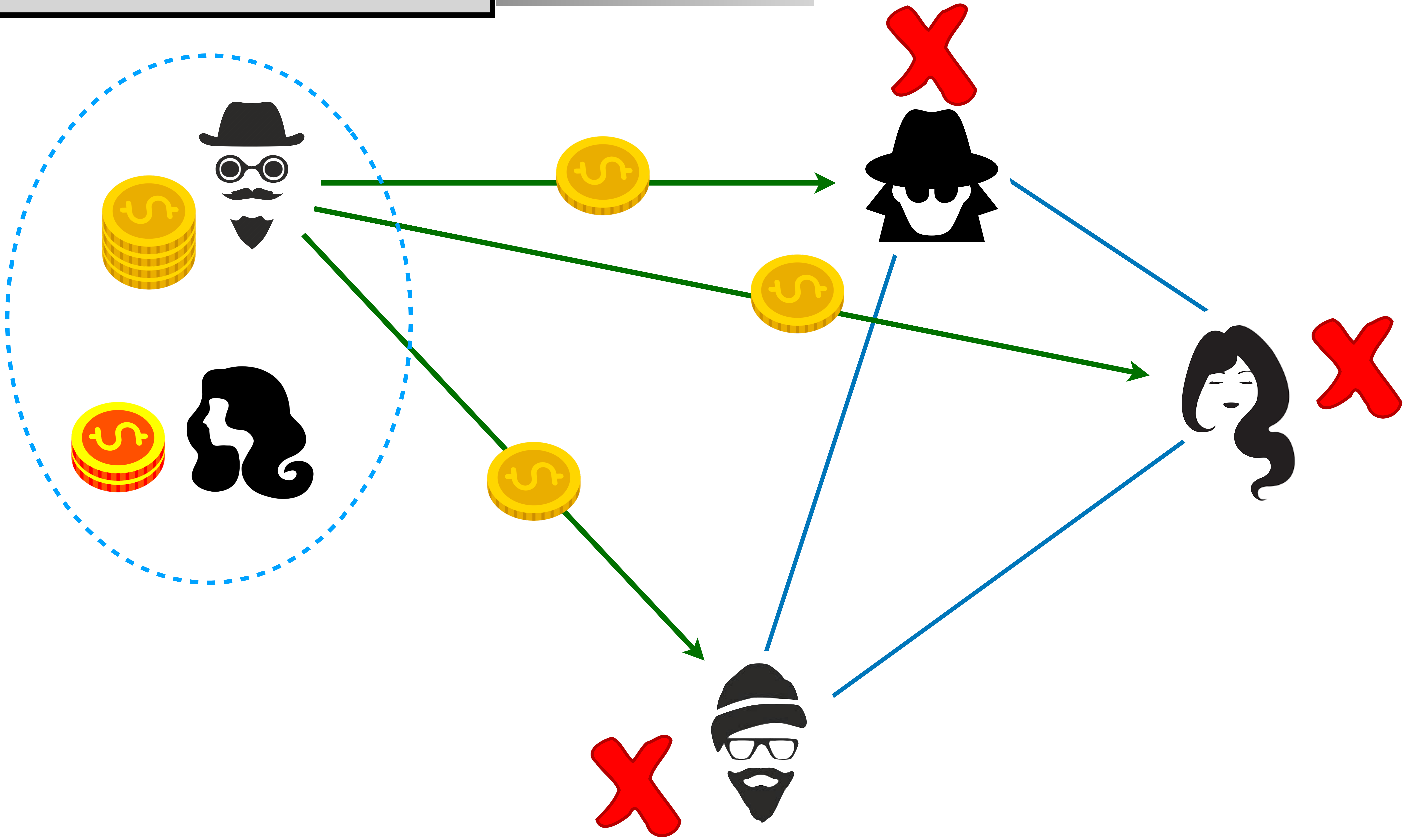
Tape sharing



Core Result: t sources, deterministic functionalities

Inner Protocol

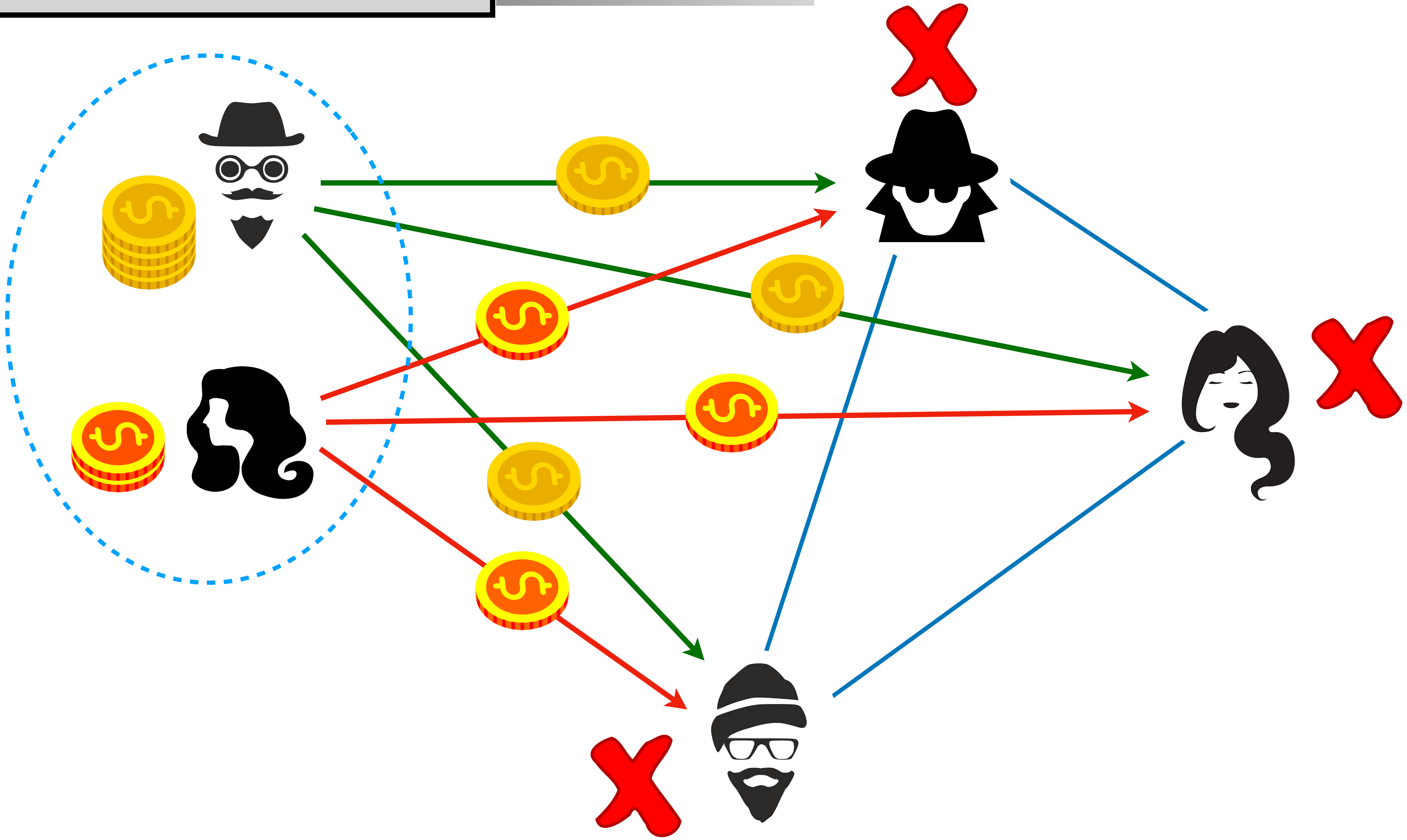
Tape sharing



Core Result: t sources, deterministic functionalities

Inner Protocol

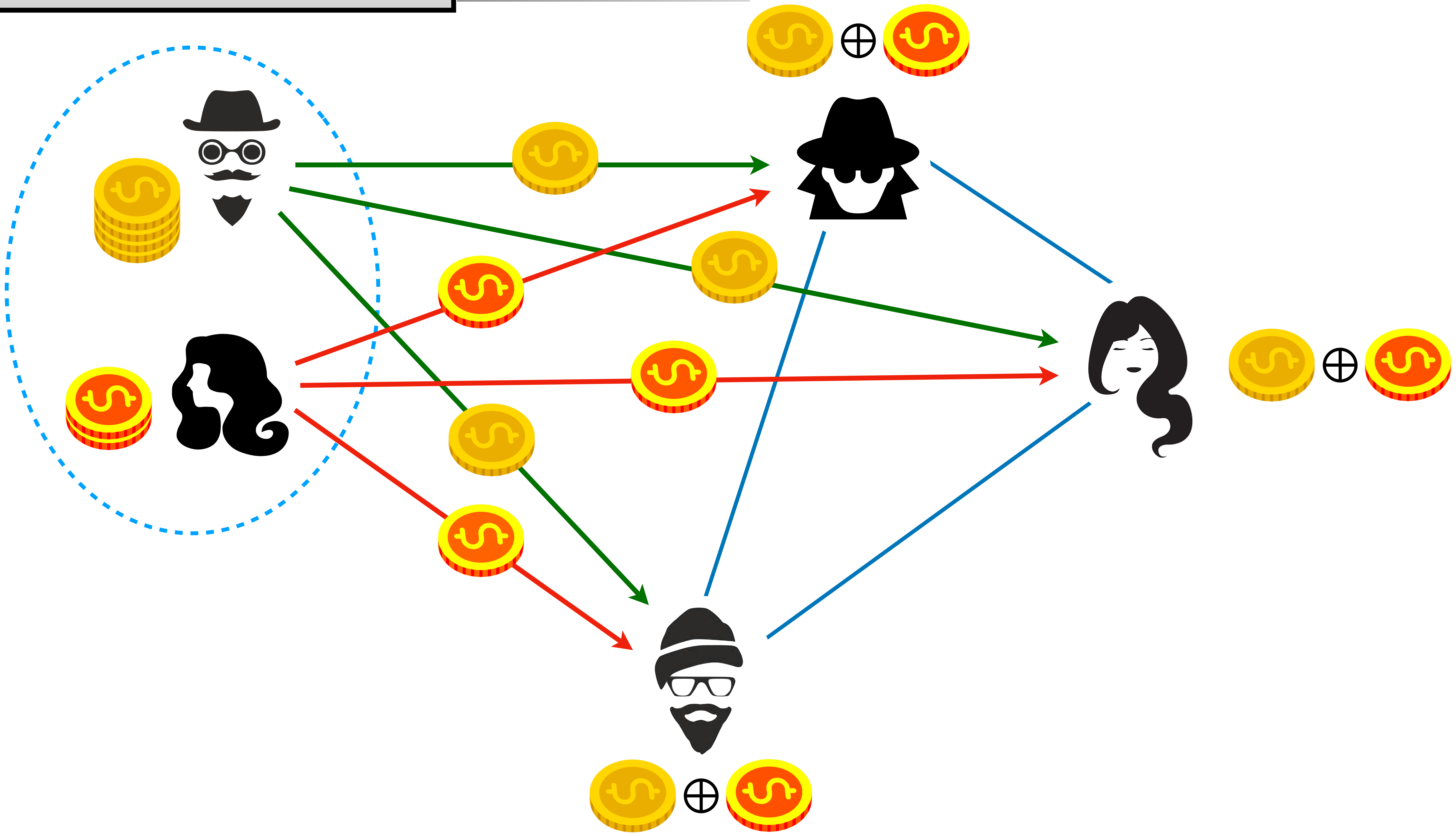
Tape sharing



Core Result: t sources, deterministic functionalities

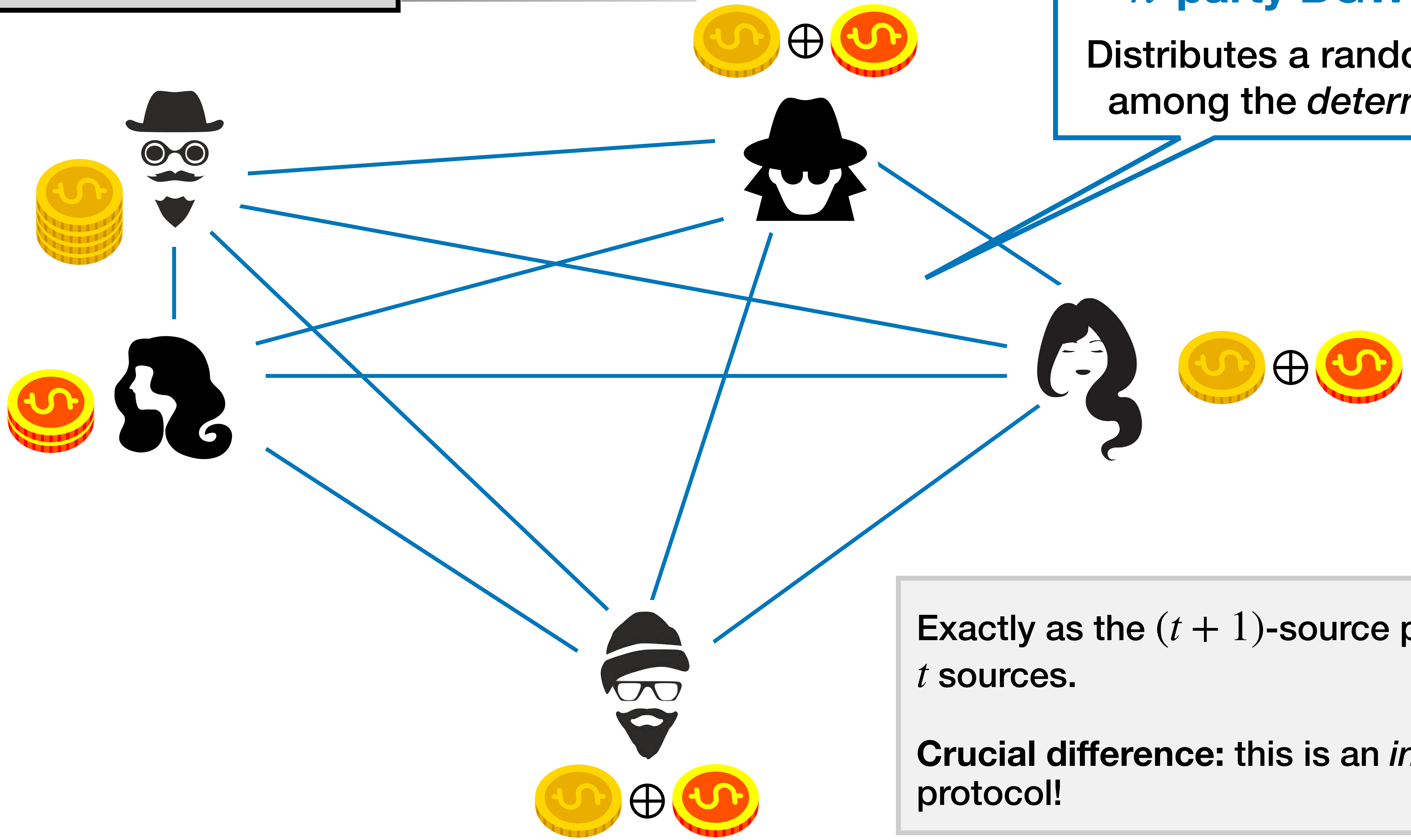
Inner Protocol

Tape sharing



Core Result: t sources, deterministic functionalities

Inner Protocol n-party BGW



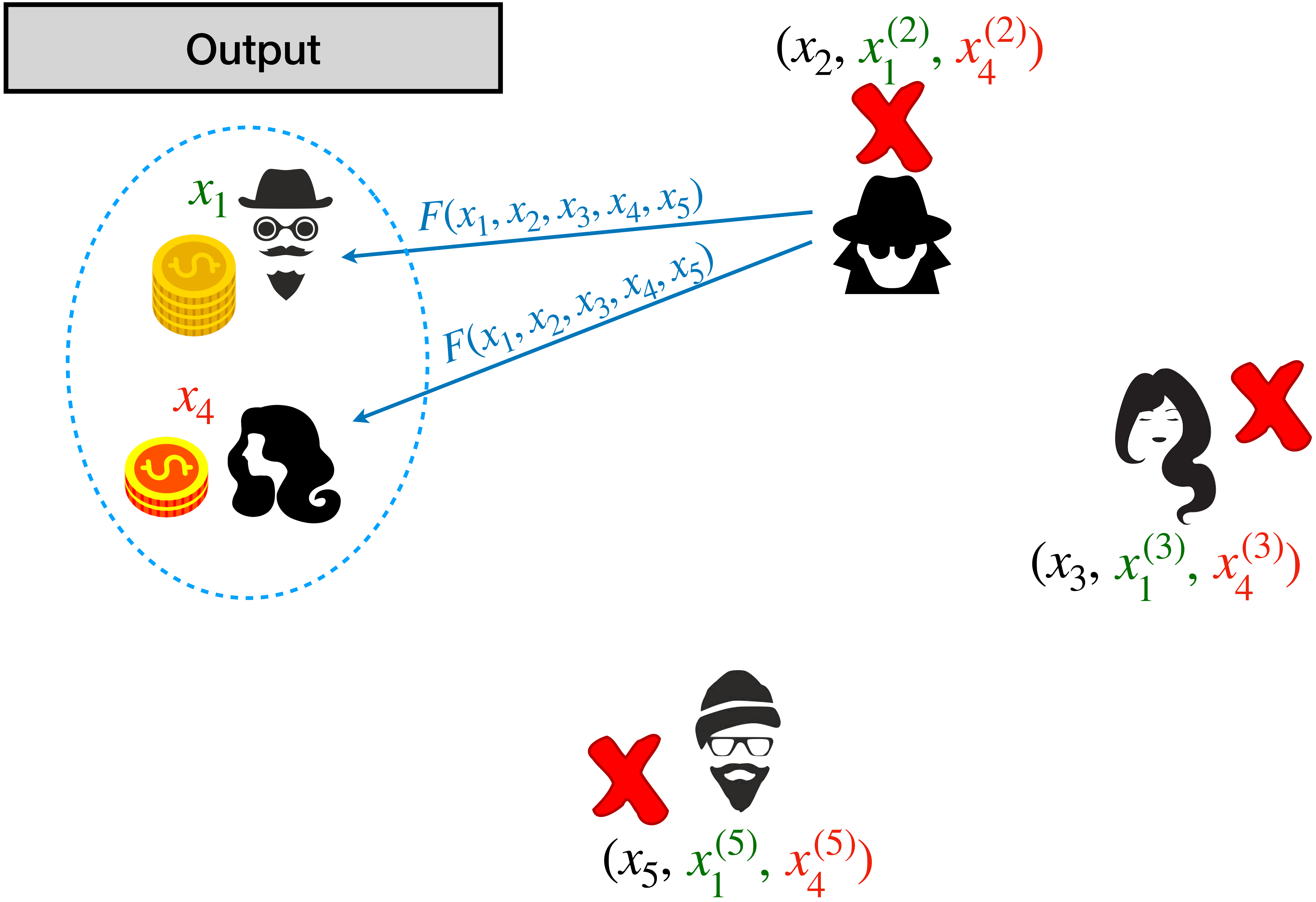
n-party BGW for $\mathscr{F}_{\text{Beaver}}$

Distributes a random Beaver triple among the *deterministic* parties

Exactly as the $(t+1)$-source protocol, but with t sources.

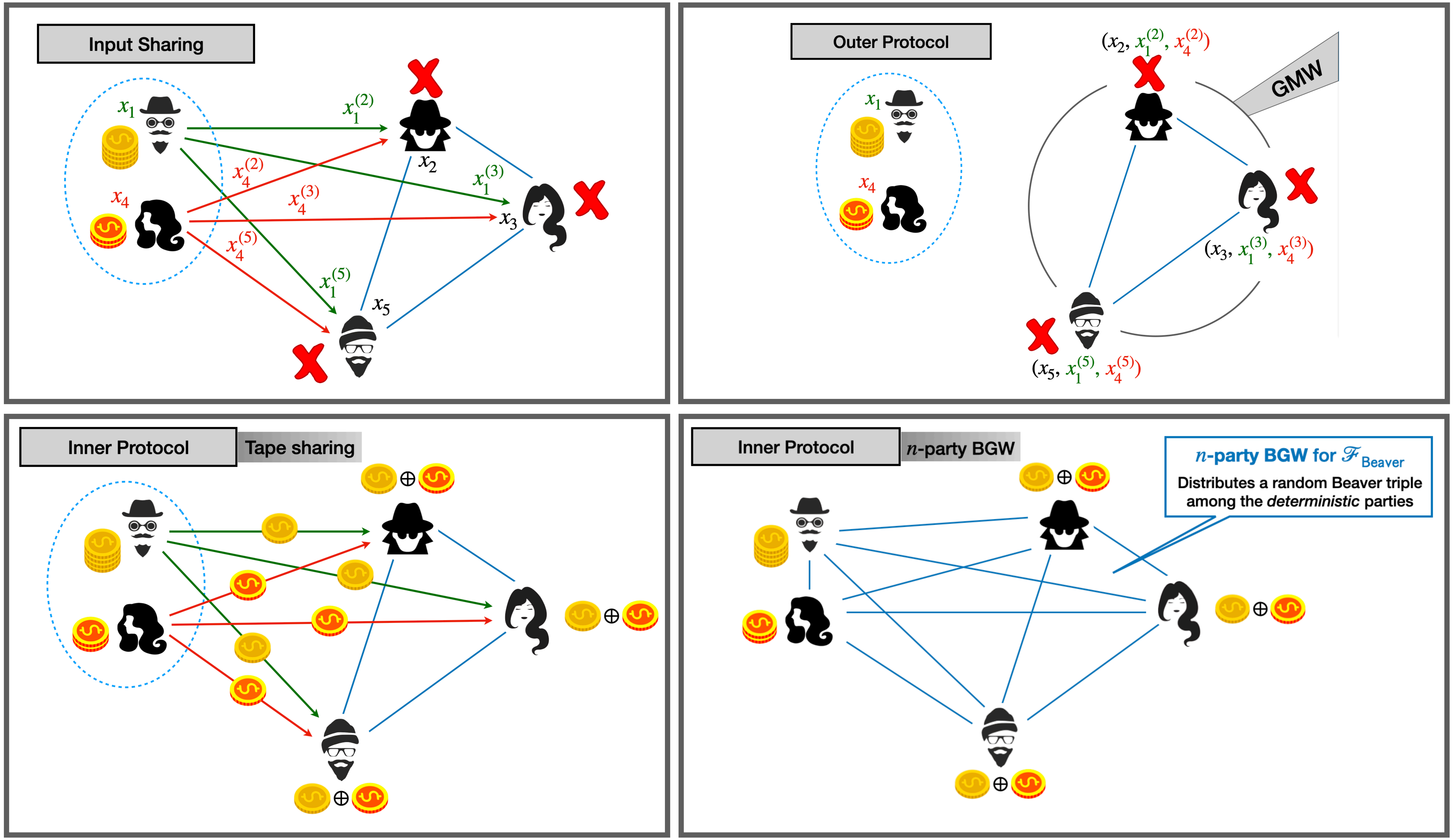
Crucial difference: this is an *input-independent* protocol!

Core Result: t sources, deterministic functionalities



Core Result: t sources, deterministic functionalities

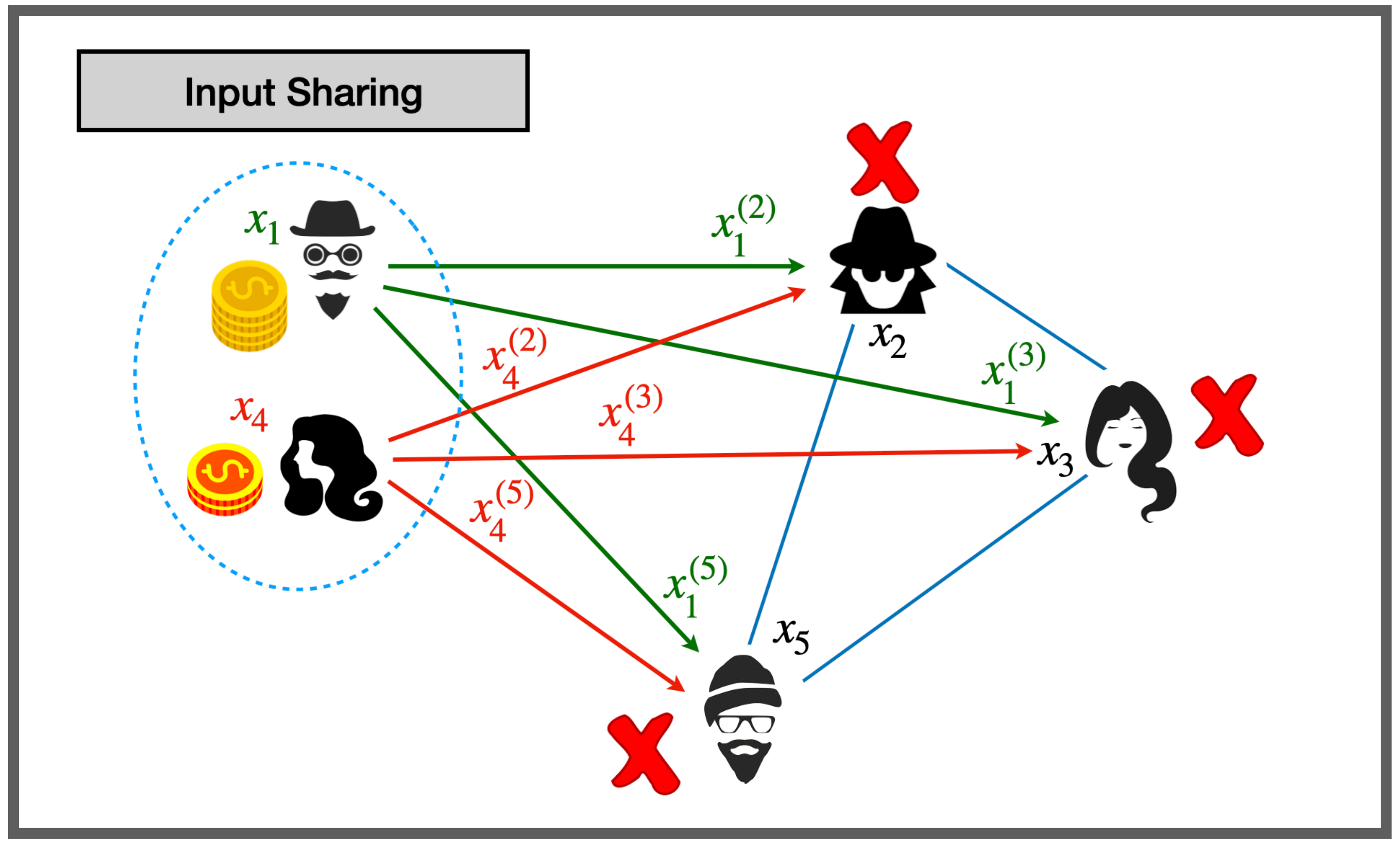
Summary of the protocol

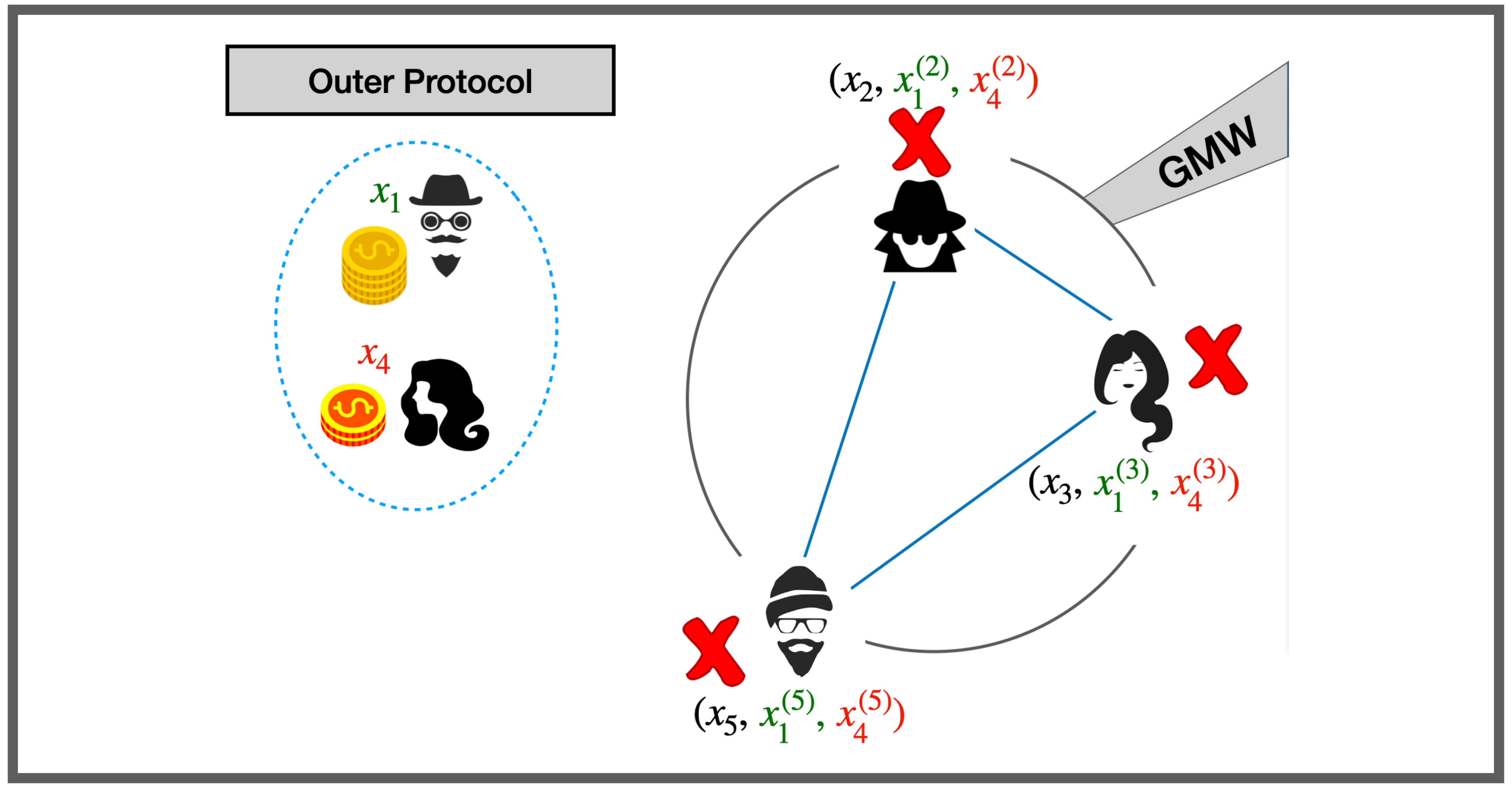


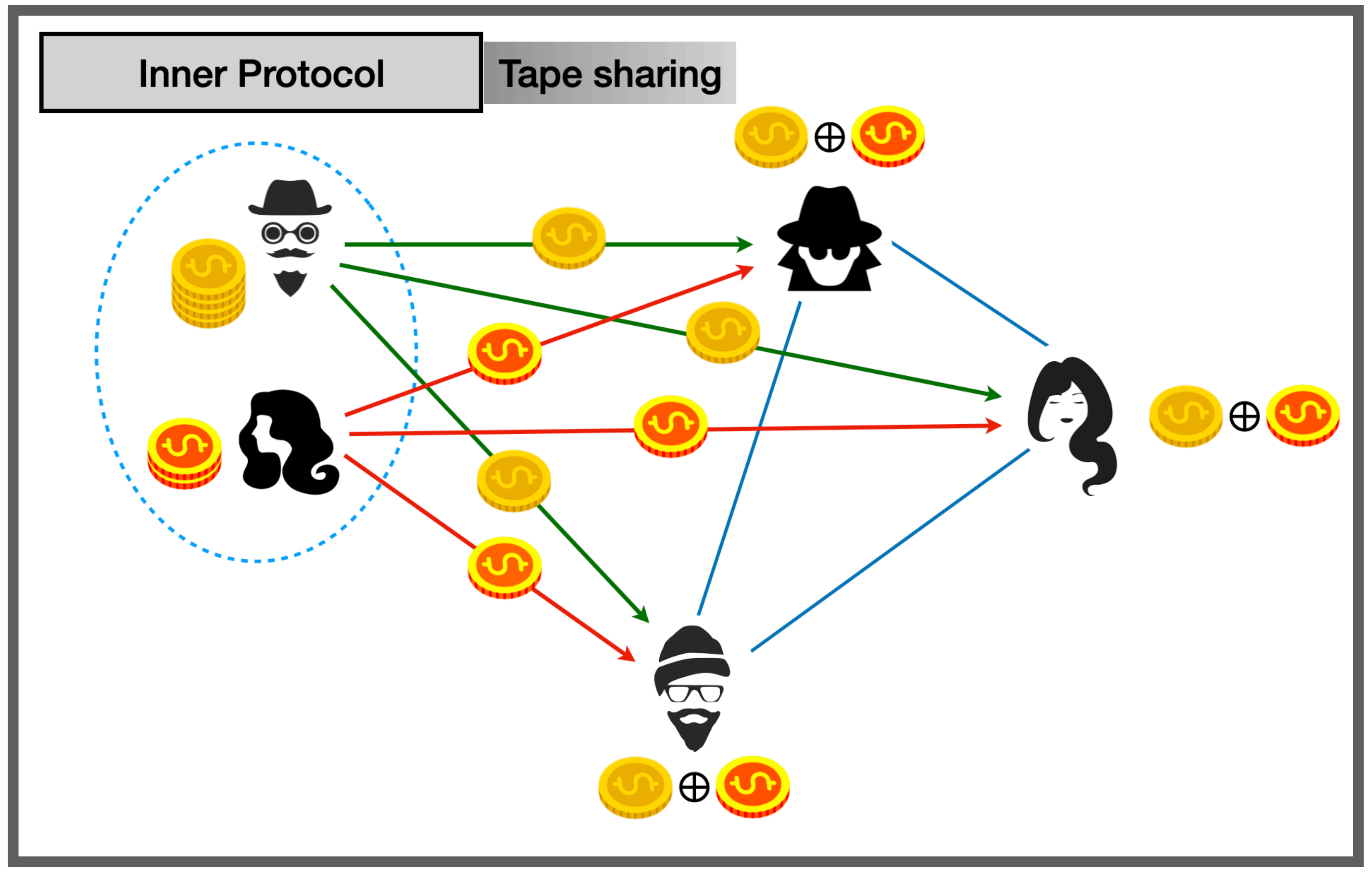
Security

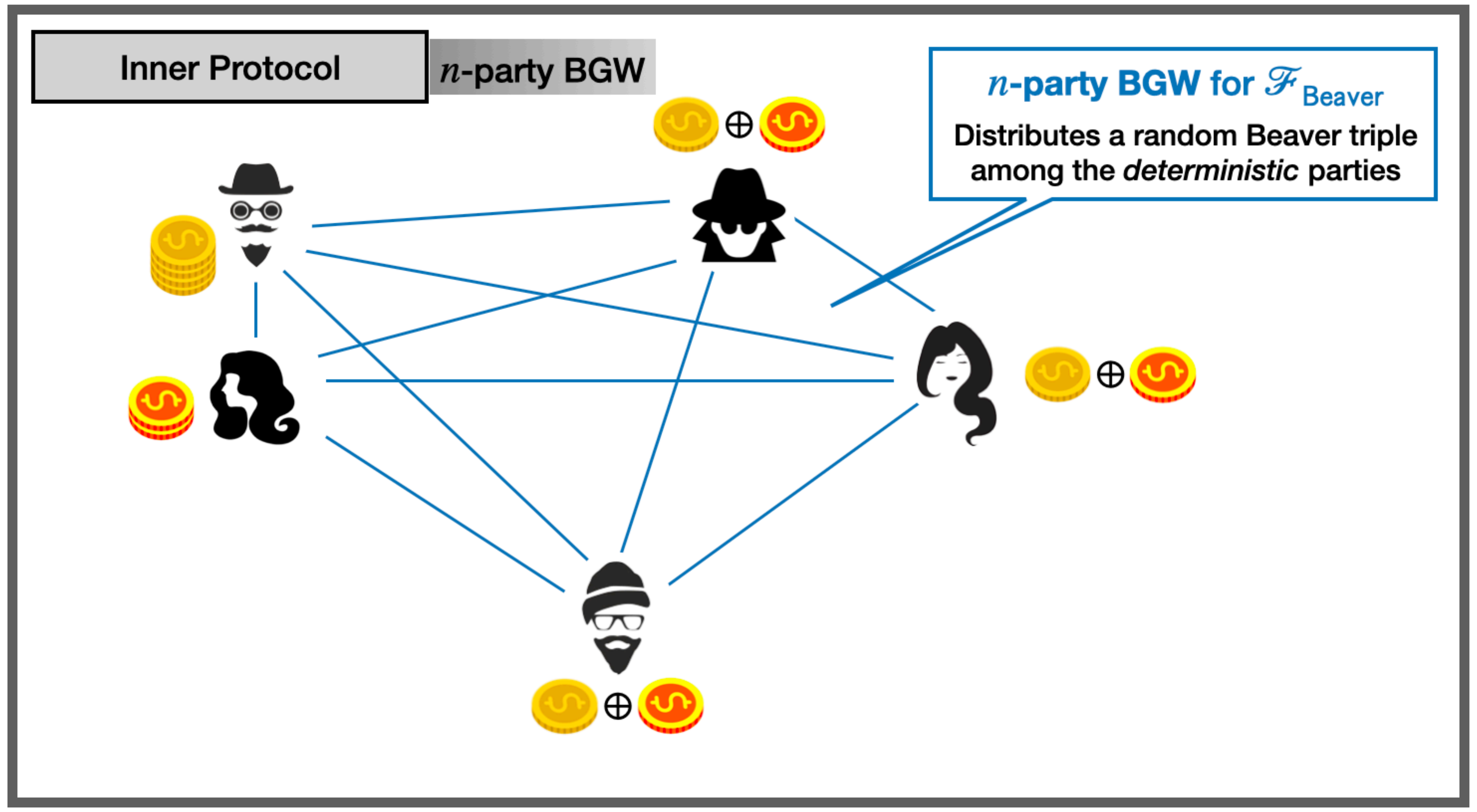
Core Result: t sources, deterministic functionalities

Summary of the protocol









Security

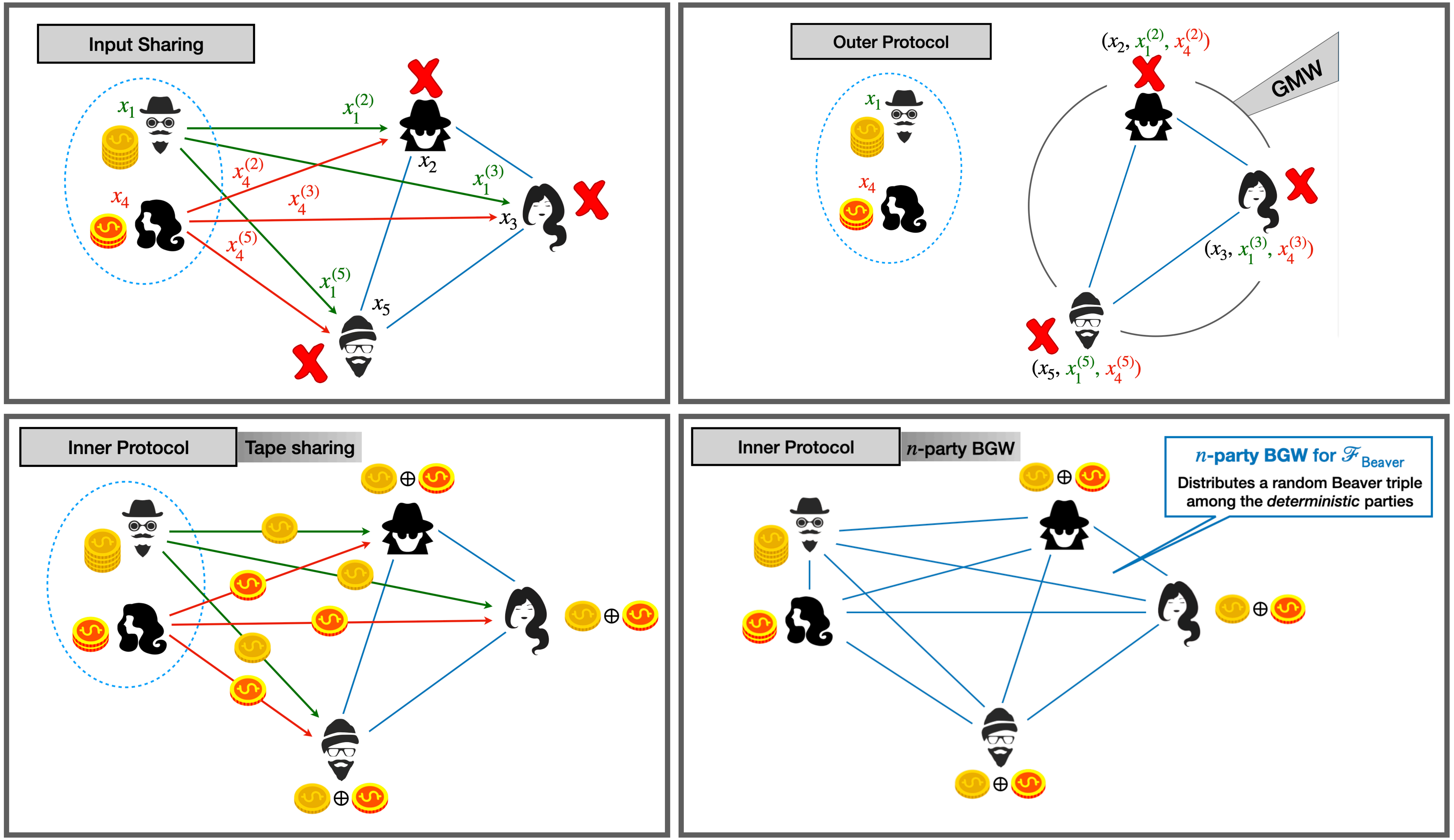
Two cases

(1) there is one honest source

(2) $\mathscr{A}$ corrupts only the sources

Core Result: t sources, deterministic functionalities

Summary of the protocol



Security

Two cases

(1) there is one honest source

(2) $\mathscr{A}$ corrupts only the sources

① Two observations

(a) $\mathscr{A}$ can't see the shared tapes

$\Longrightarrow$ BGW is secure

$\Longrightarrow$ The Beaver triples are trusted

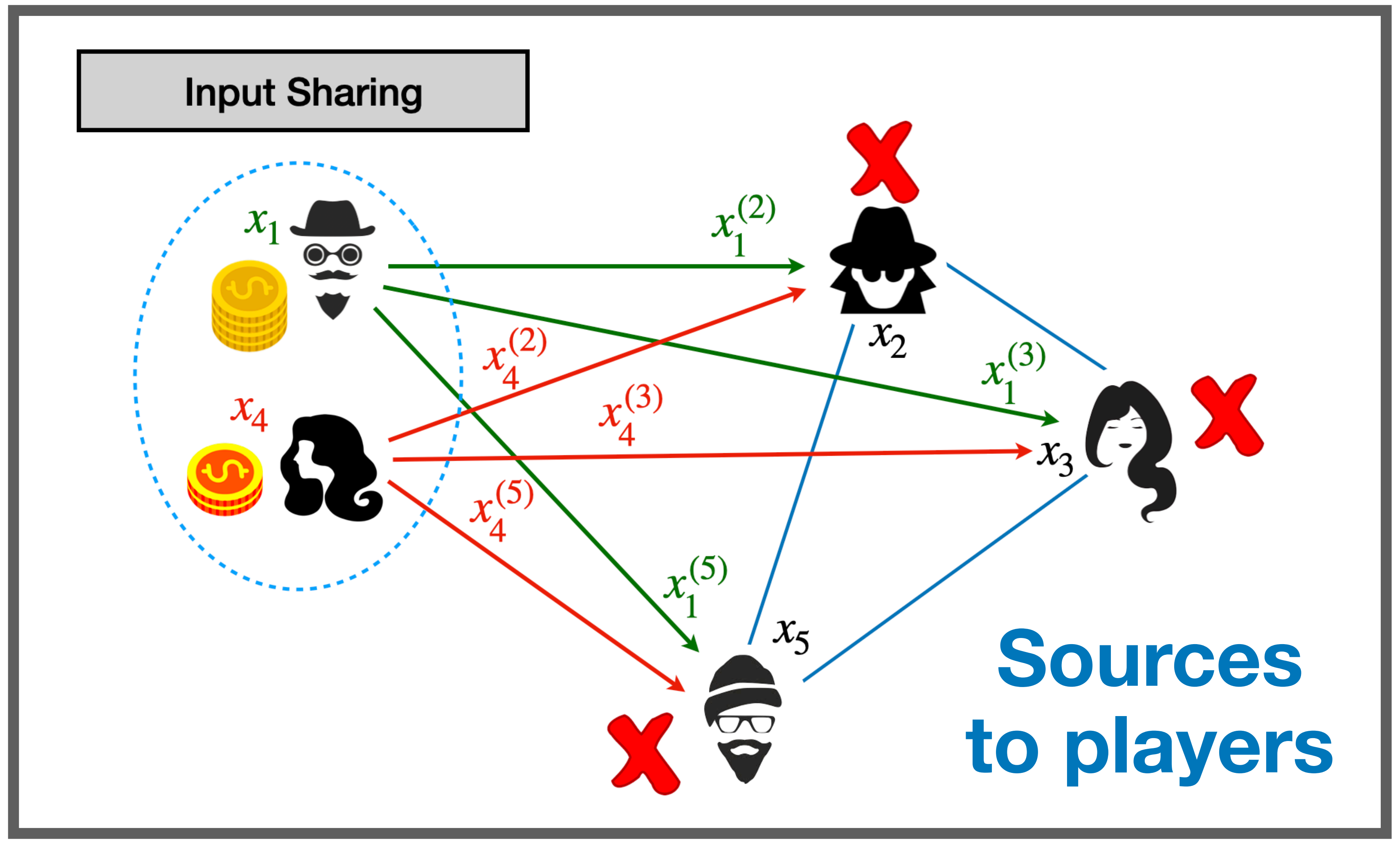
(b) $t < n/2 \implies t < n - t$

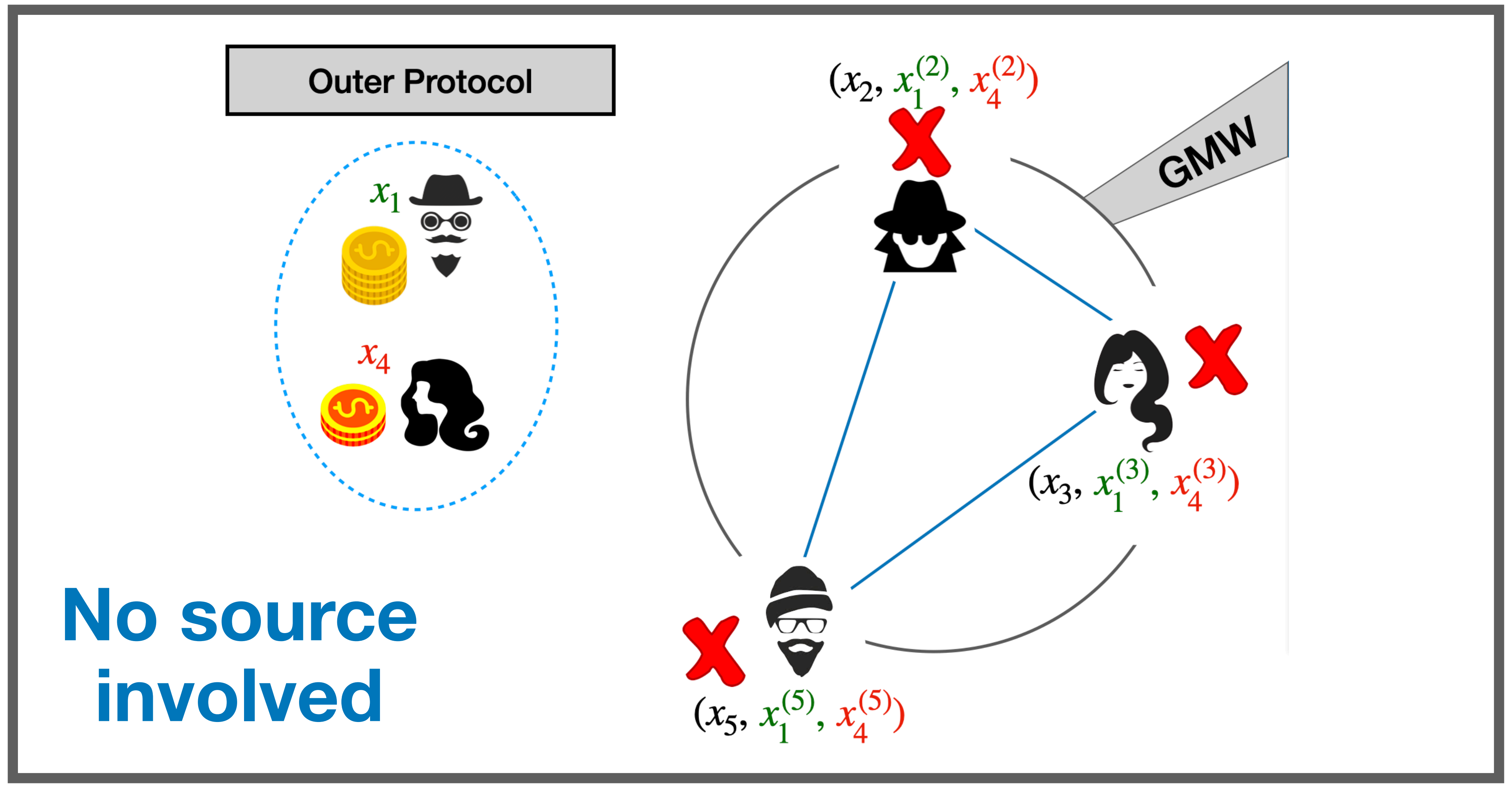
$\Longrightarrow$ there is one honest player

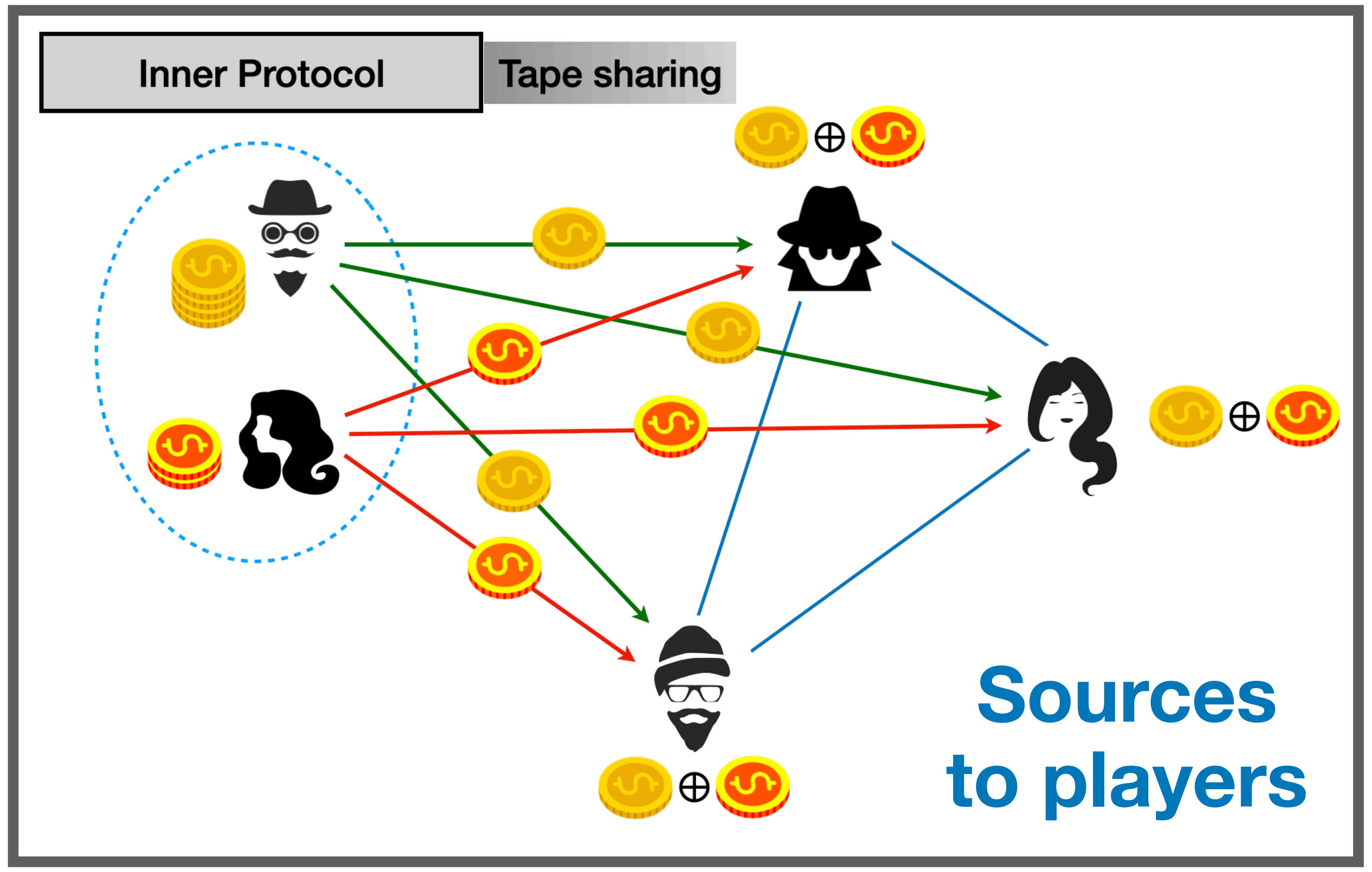
$\Longrightarrow$ GMW is secure ✓

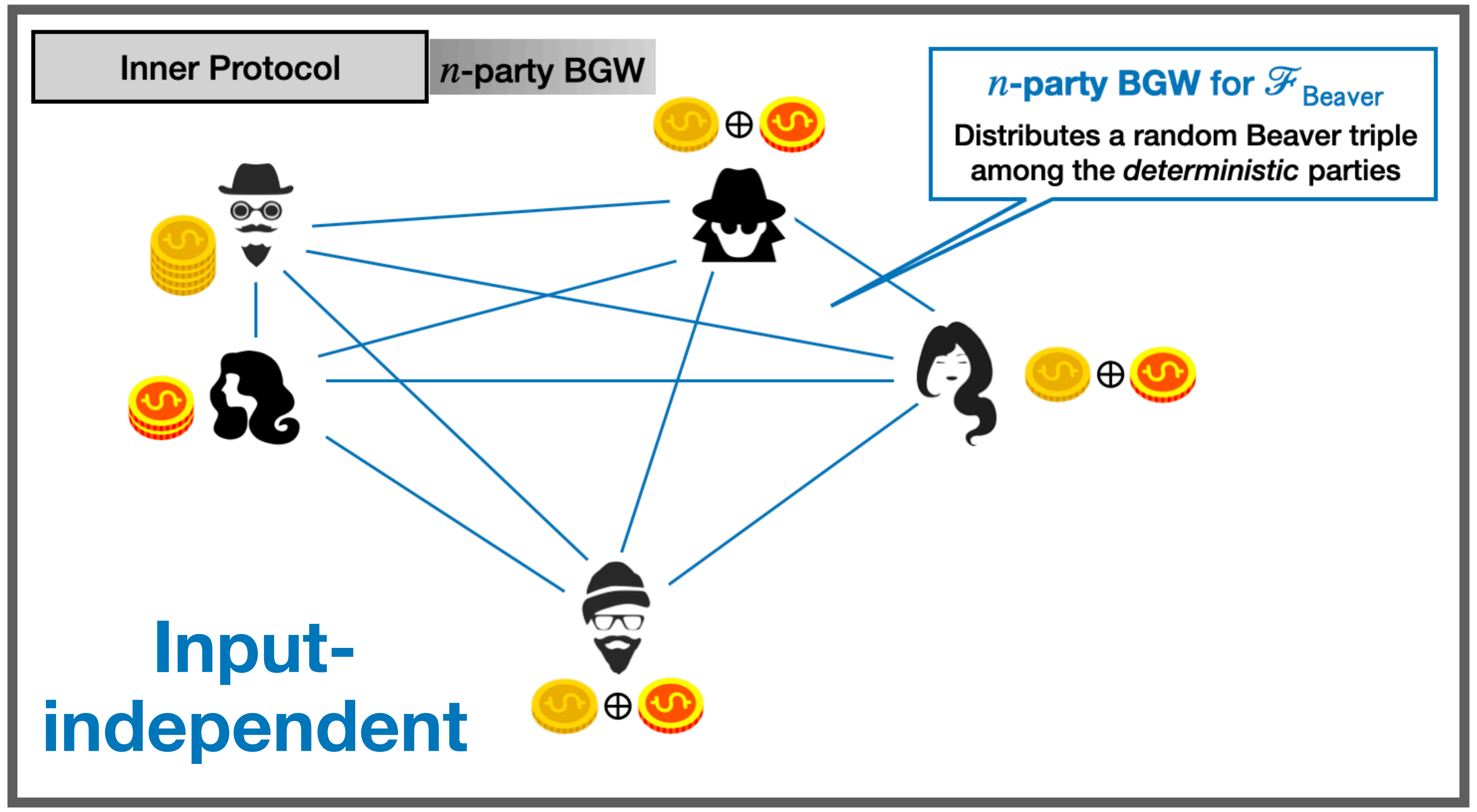
Core Result: t sources, deterministic functionalities

Summary of the protocol









Security

Two cases

(1) there is one honest source

(2) $\mathscr{A}$ corrupts only the sources

1 Two observations

(a) $\mathscr{A}$ can't see the shared tapes

$\Longrightarrow$ BGW is secure

$\Longrightarrow$ The Beaver triples are trusted

(b) $t < n/2 \implies t < n - t$

$\Longrightarrow$ there is one honest player

$\Longrightarrow$ GMW is secure ✓

2 One observation

no source ever sees an input-dependent message, beyond the output! ✓

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Motivation & Previous Work

- AND is a basic building block of MPC, together with XOR (for which we already have tight — trivial — bounds)
- The randomness complexity of 1-private n-party AND has been studied in previous works
- Most recent result [TCC:KOPRTV'19]: AND can be computed using 8 bits (and two sources)

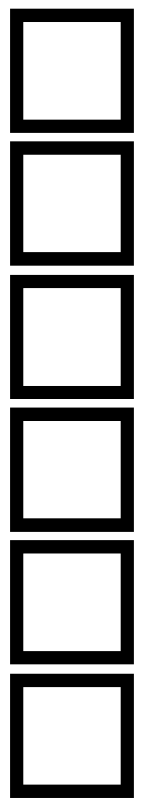
Setting

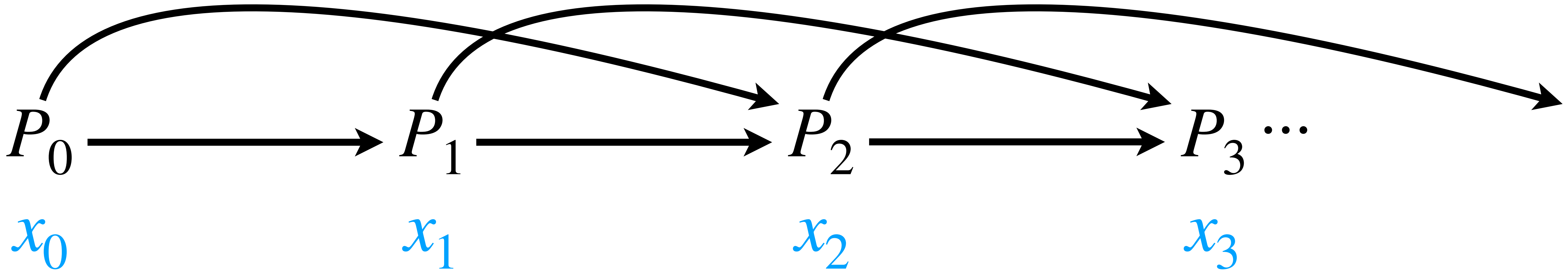
- n parties $(P_0, \cdots, P_{n-1})$ with respective inputs $(x_0, \cdots, x_{n-1})$
- Output: $\wedge_{i=0}^{n-1} x_i$
- At most one corrupted party (= no collusion)

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase

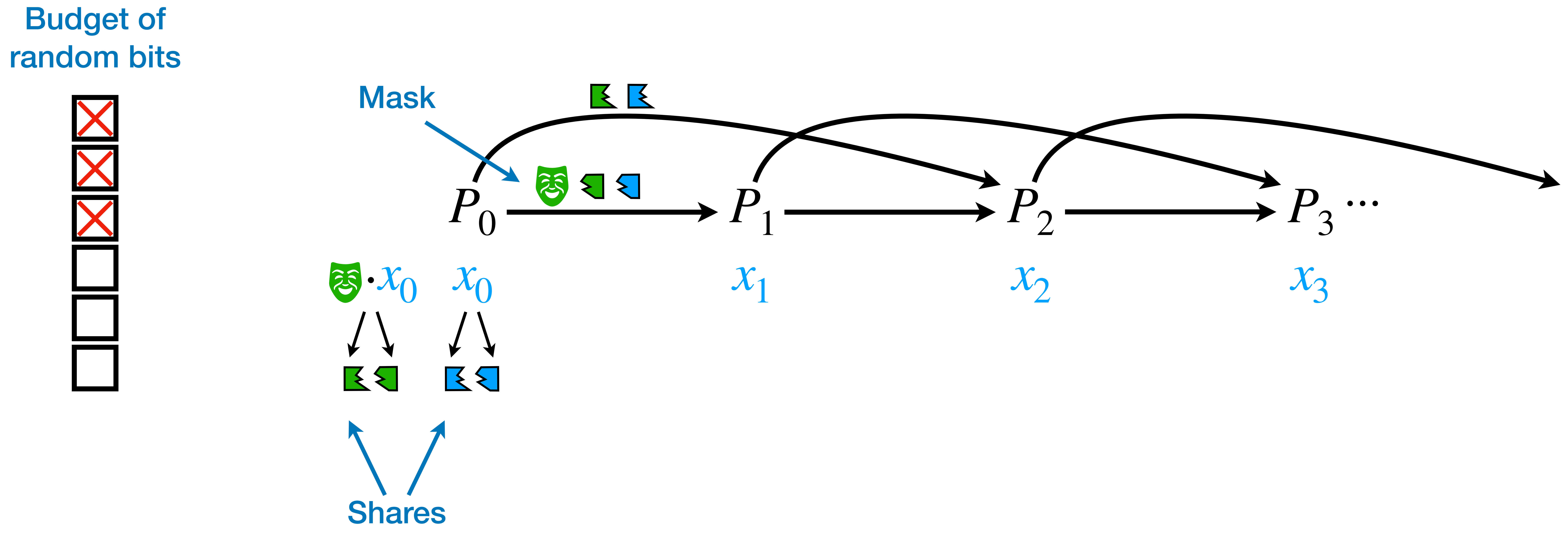
Budget of random bits





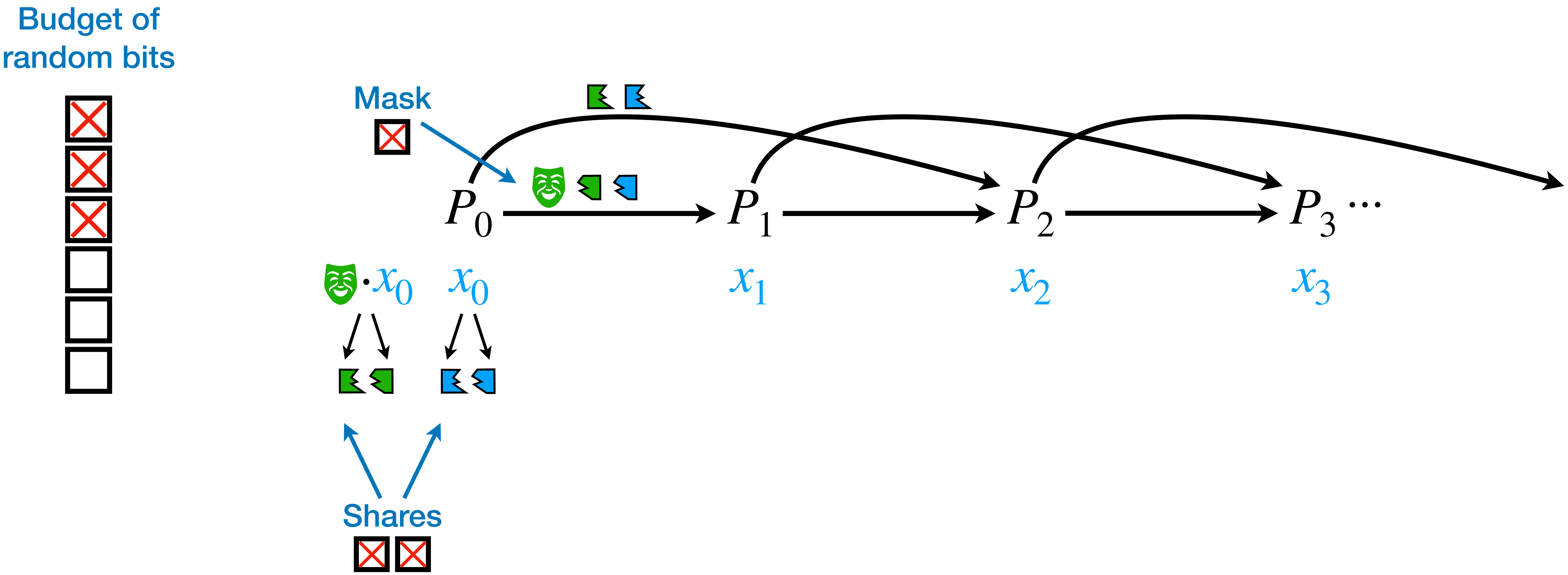
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



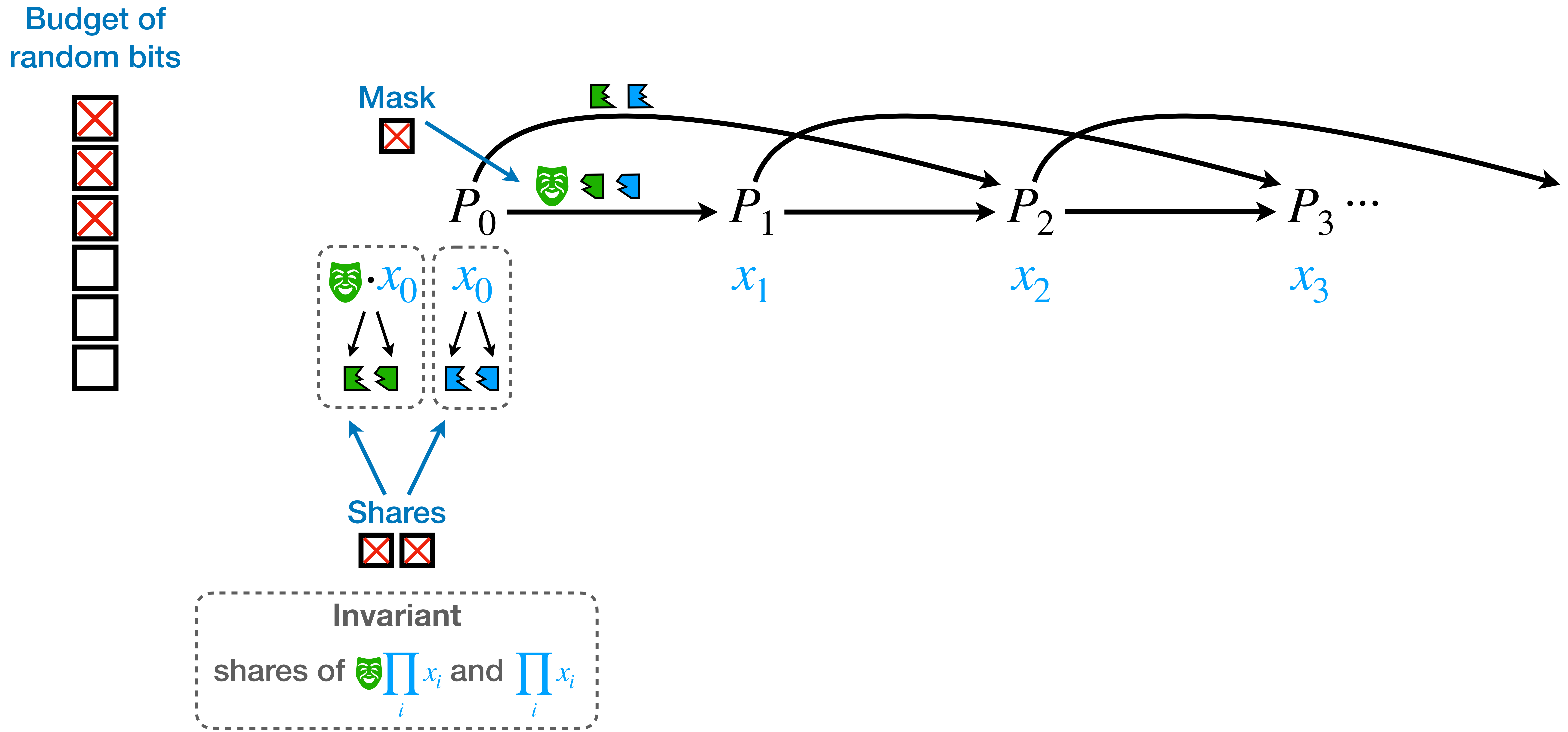
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



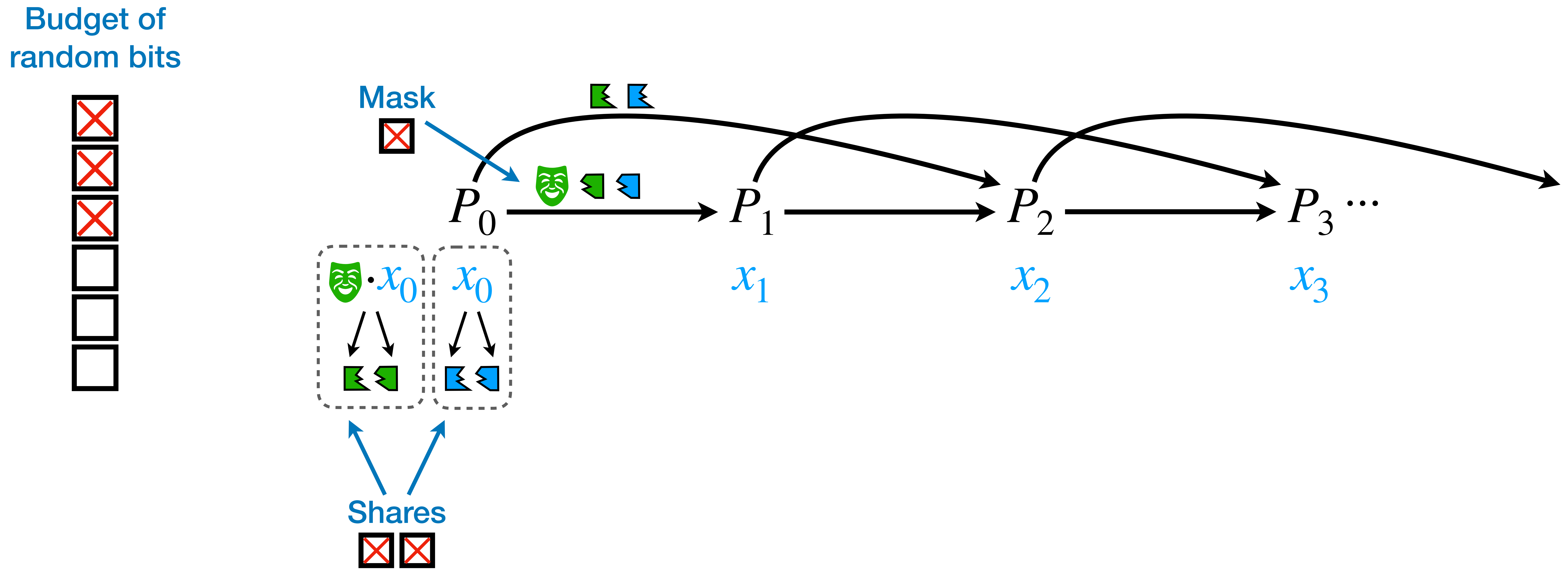
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



Invariant

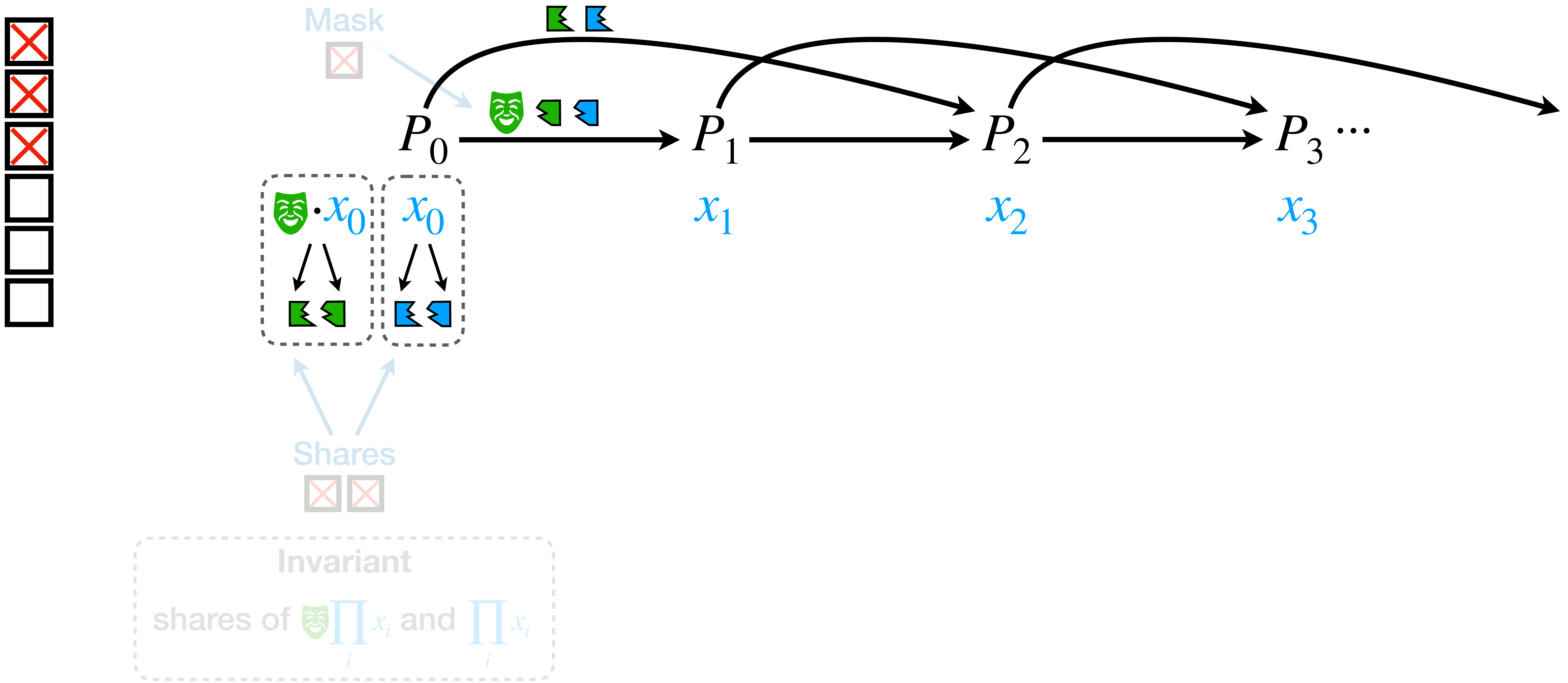
shares of $\prod_i x_i$ and $\prod_i x_i$

We *propagate* the invariant: throughout the computation, P_{i-1} and P_i will hold shares of these products.

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase

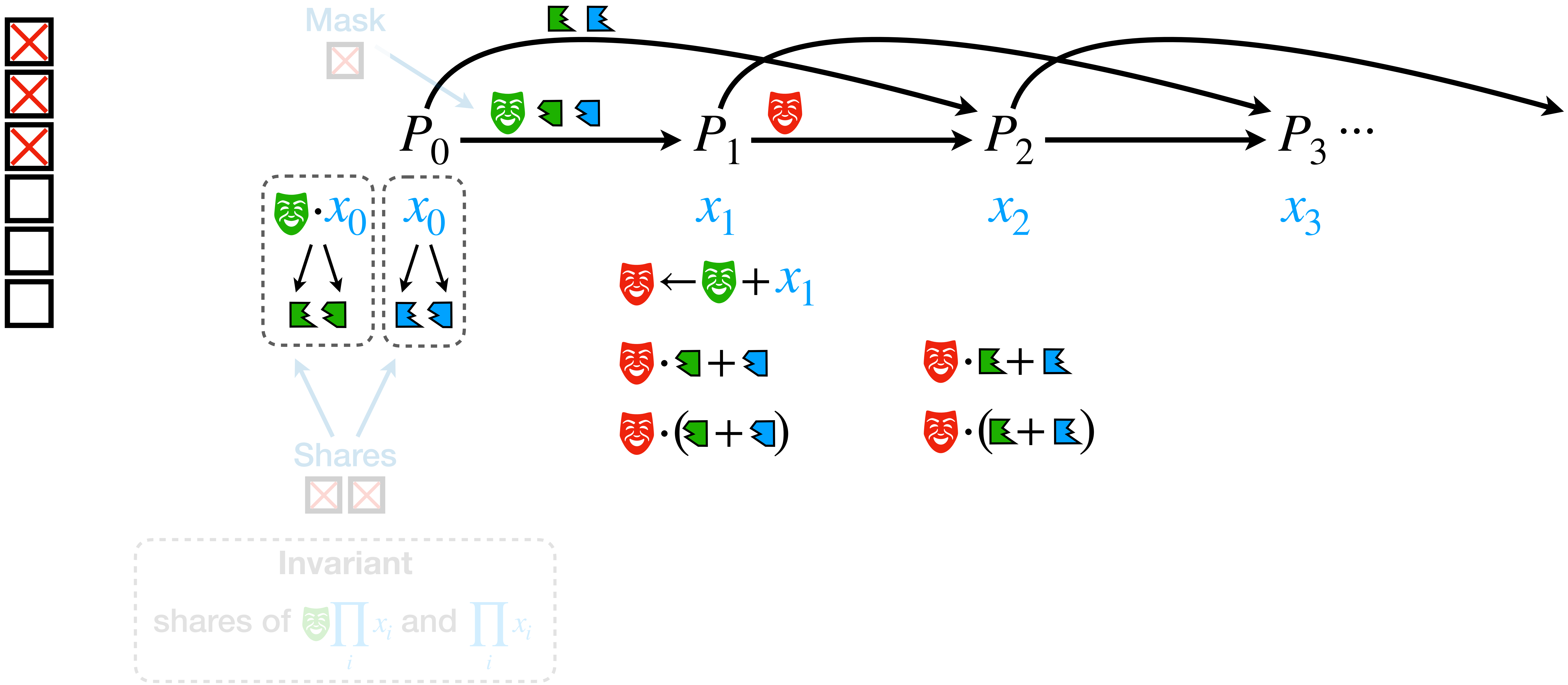
Budget of random bits



Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

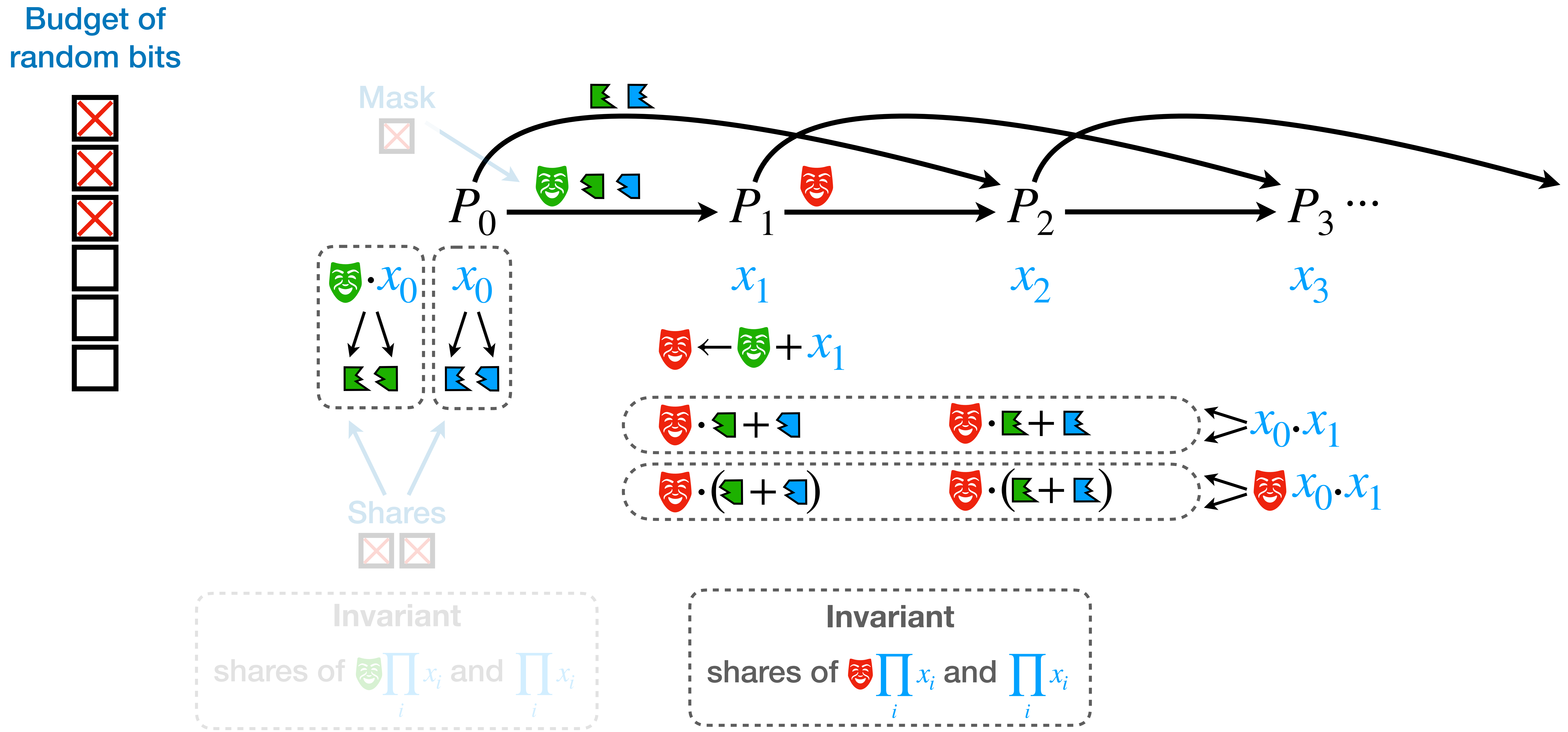
Main Phase

Budget of random bits



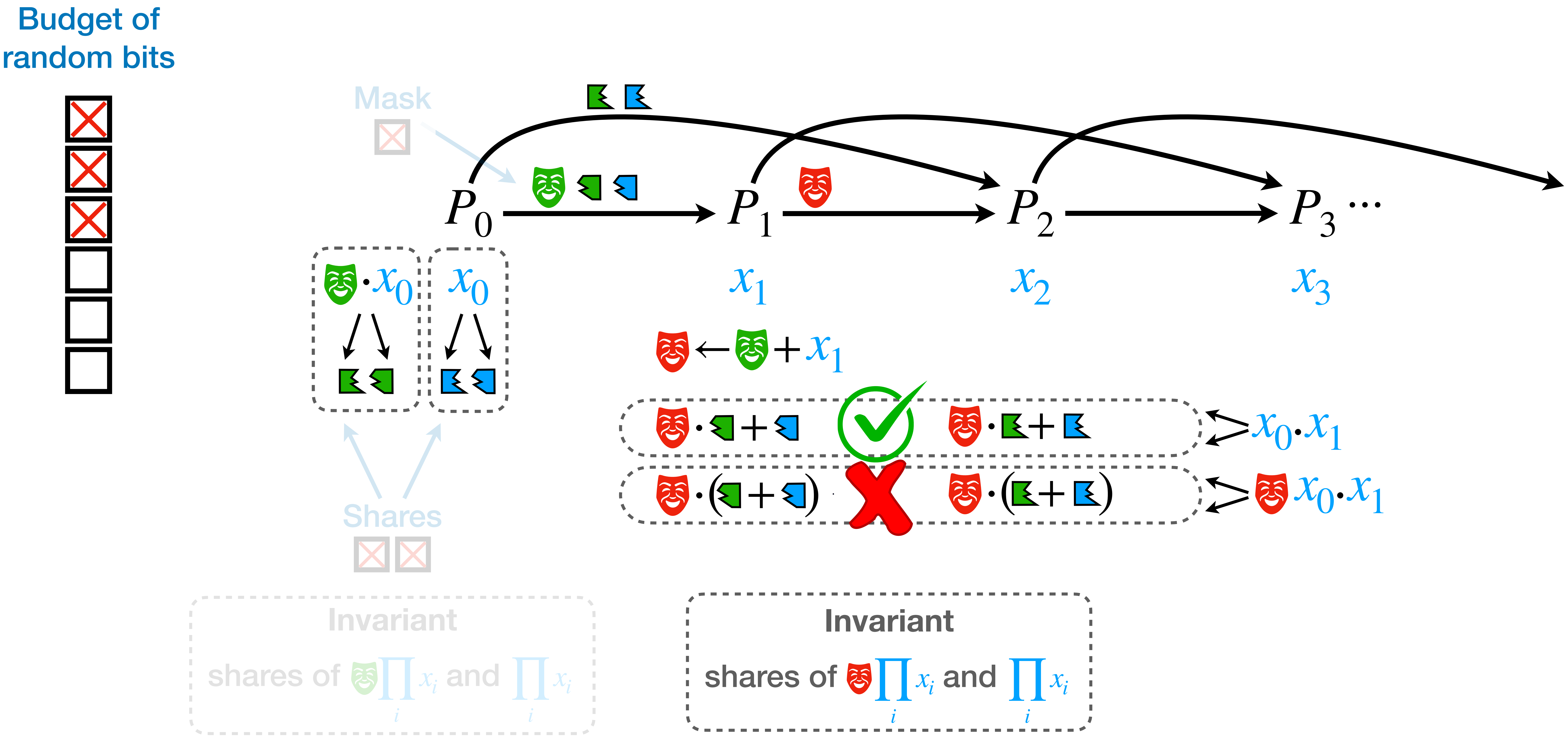
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

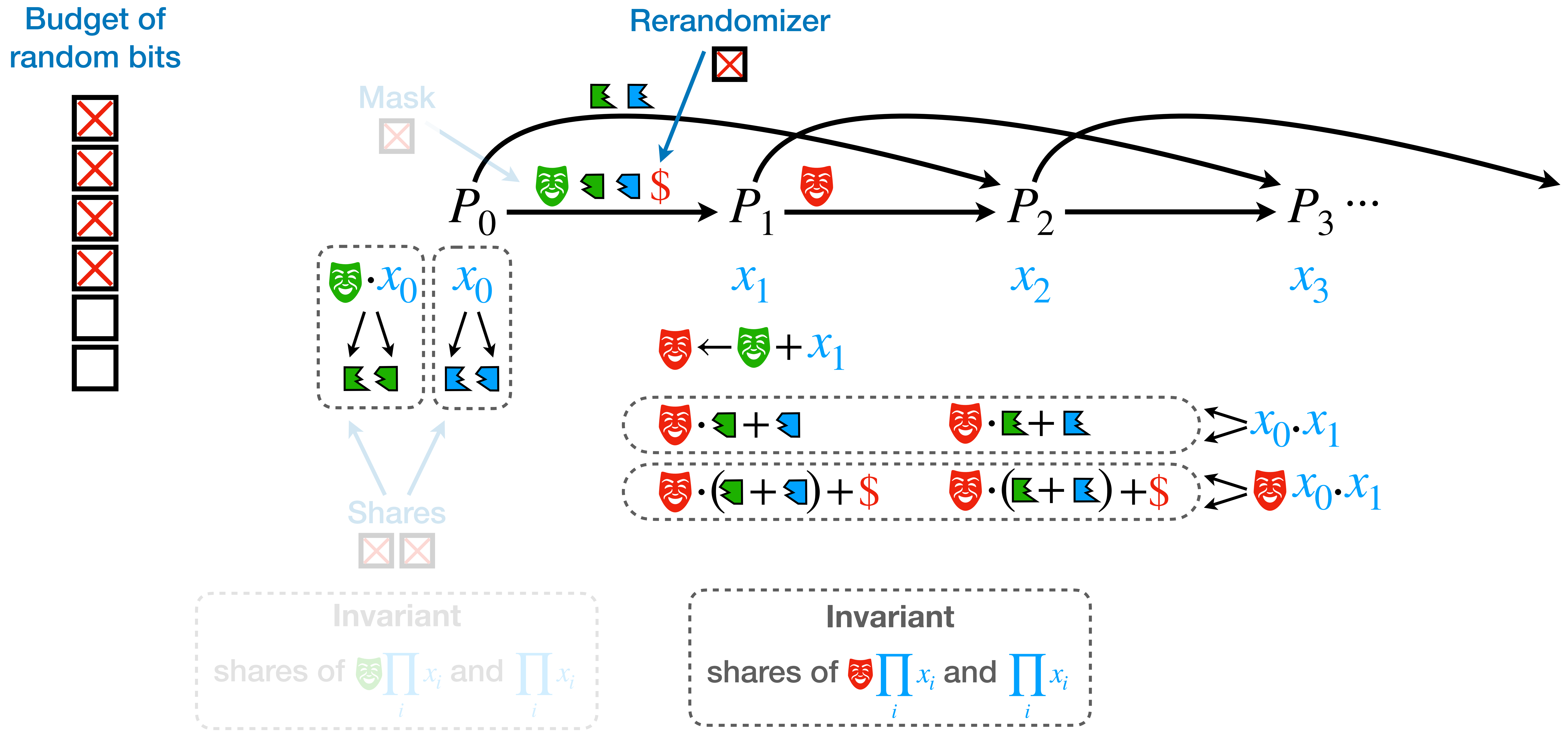
Main Phase



Shares of $x_0 \cdot x_1$ are not uniform! (Biased toward 0)

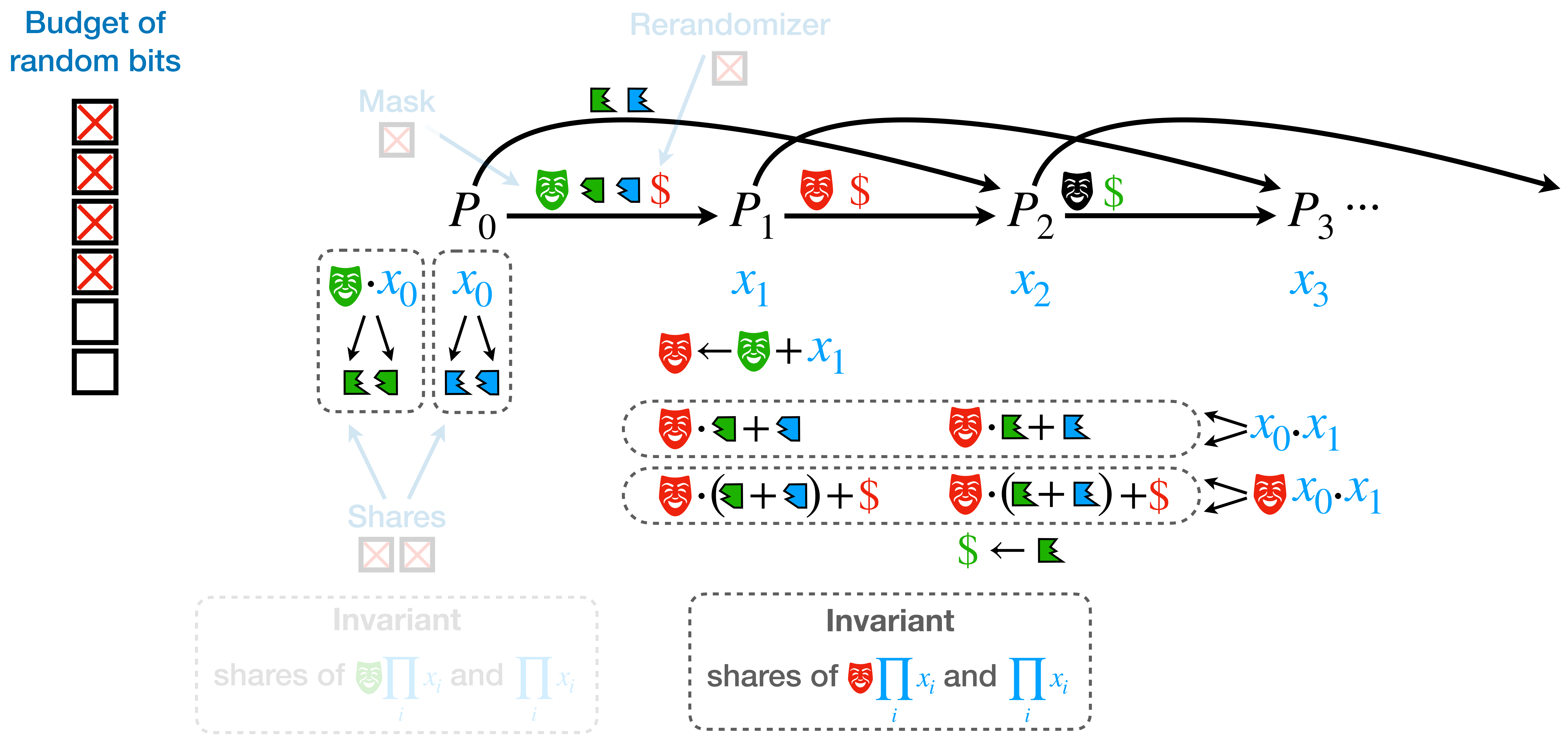
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



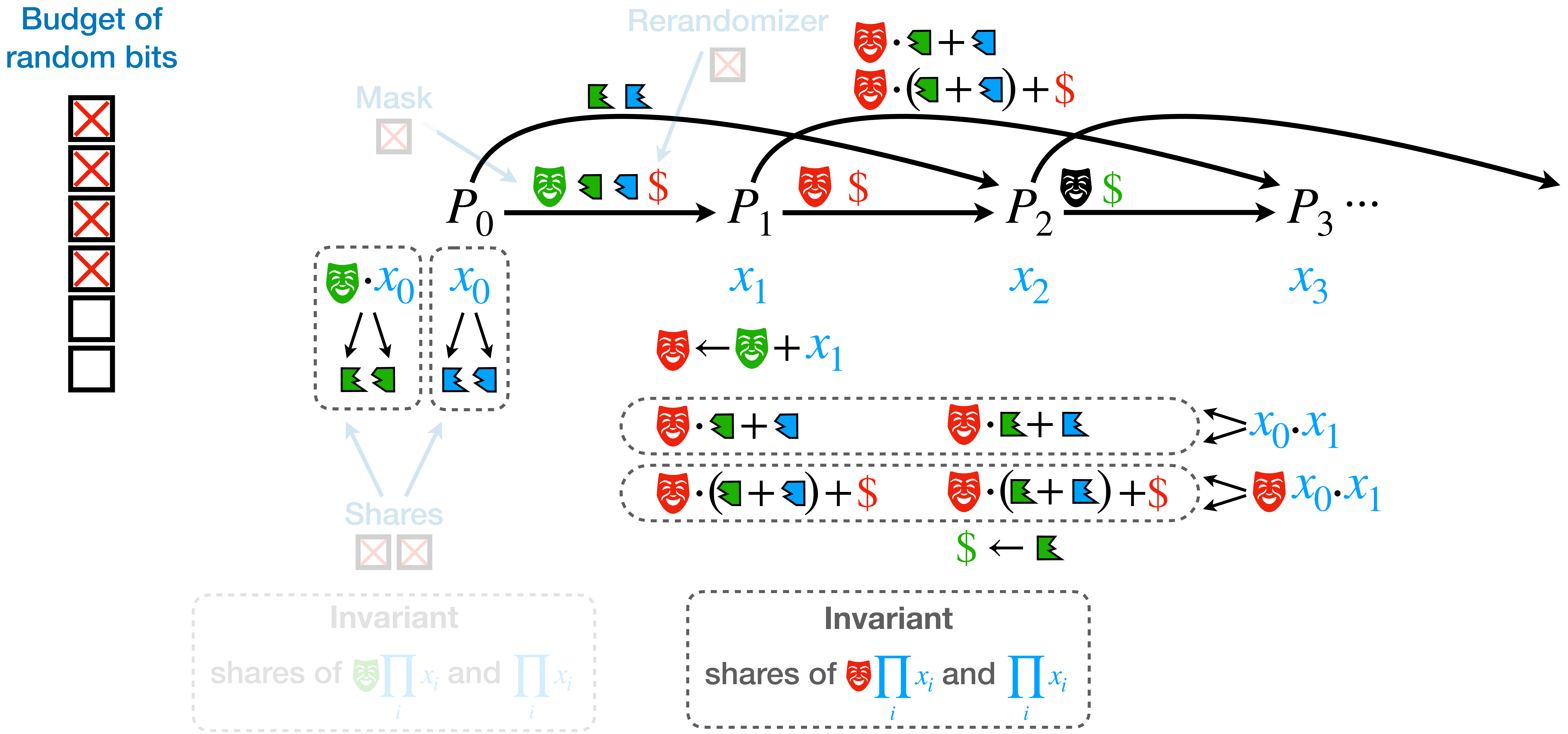
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



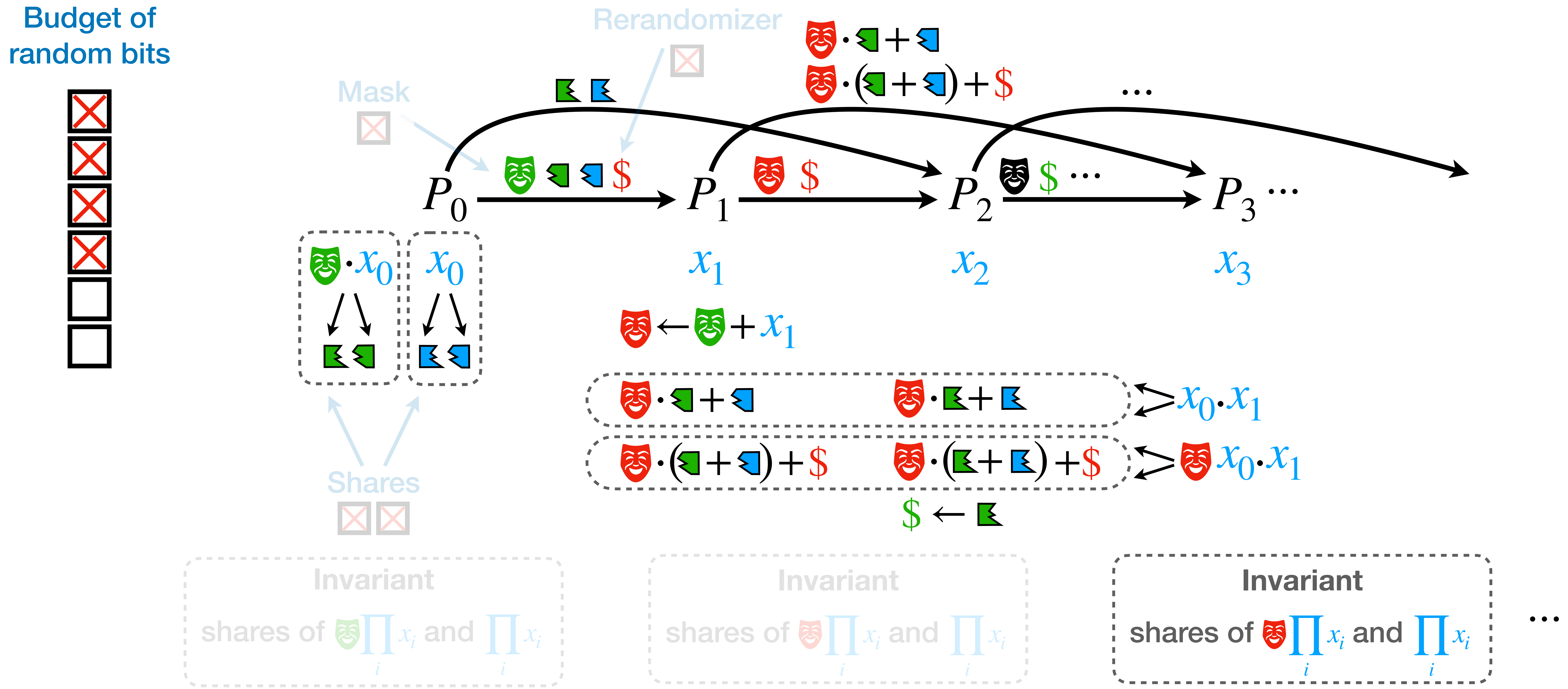
Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Main Phase



Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

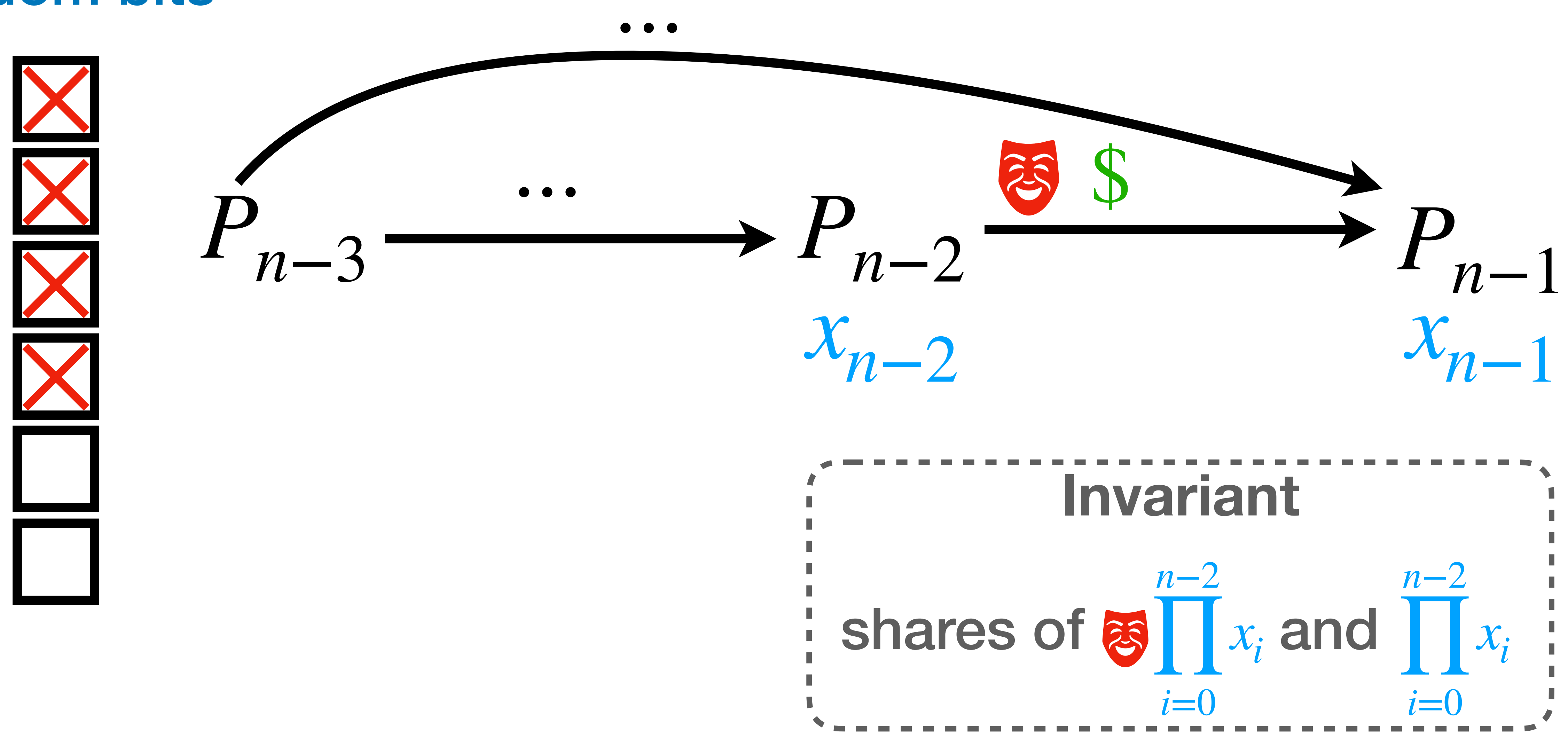
Main Phase



Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

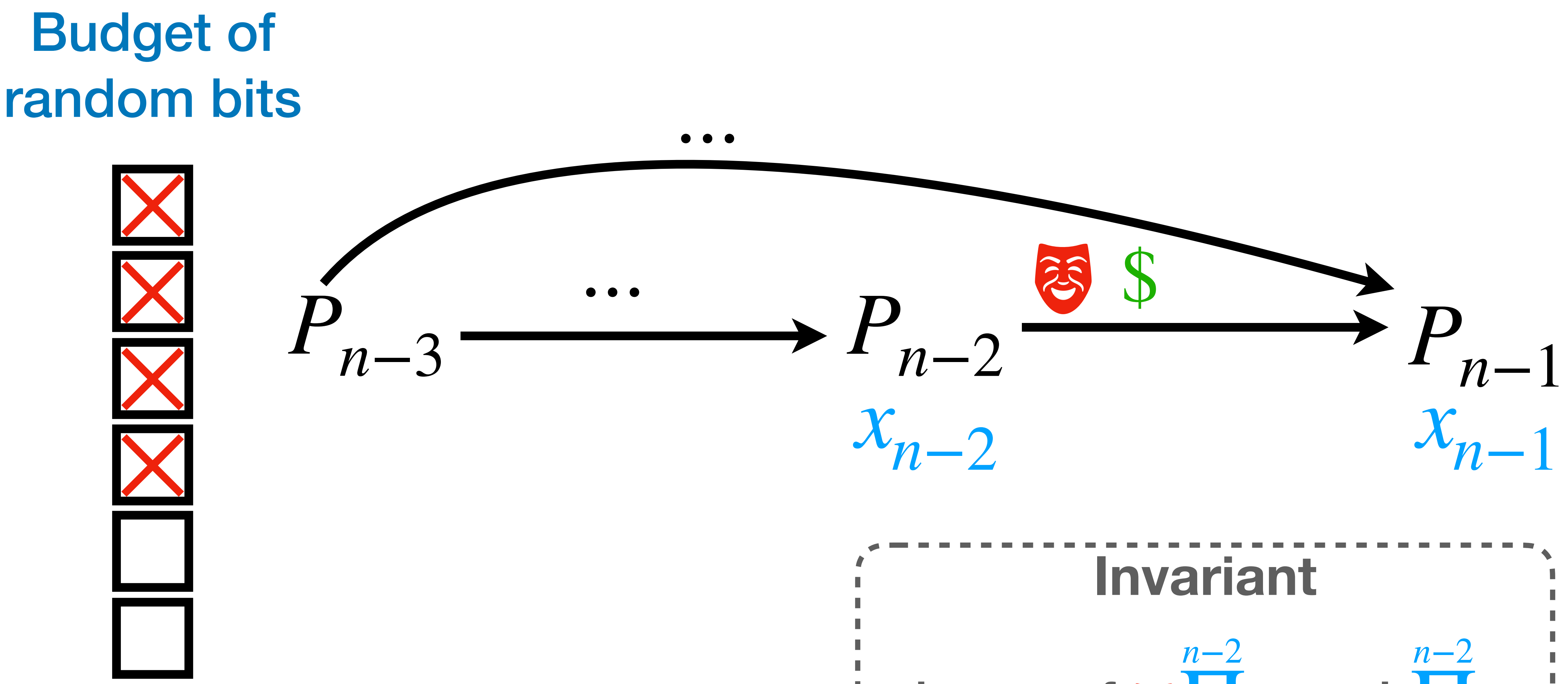
Output Phase

Budget of random bits



Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Output Phase



Invariant

shares of $\prod_{i=0}^{n-2} x_i$ and $\prod_{i=0}^{n-2} x_i$

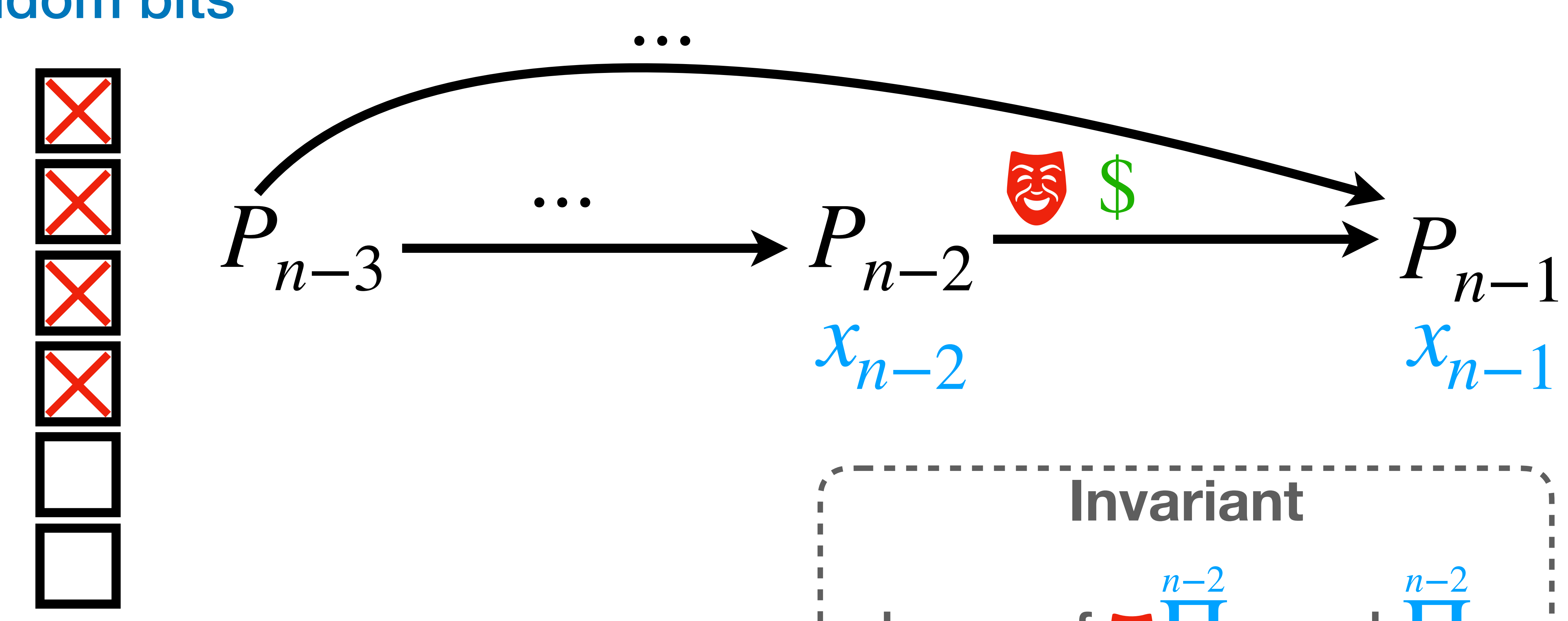
Output

$$\begin{cases} 0 & \text{if } x_{n-1} = 0 \\ \prod_{i=0}^{n-1} x_i & \text{if } x_{n-1} = 1 \end{cases}$$

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Output Phase

Budget of random bits



Output

$$\begin{cases} 0 & \text{if } x_{n-1} = 0 \\ \prod_{i=0}^{n-1} x_i & \text{if } x_{n-1} = 1 \end{cases}$$

Invariant

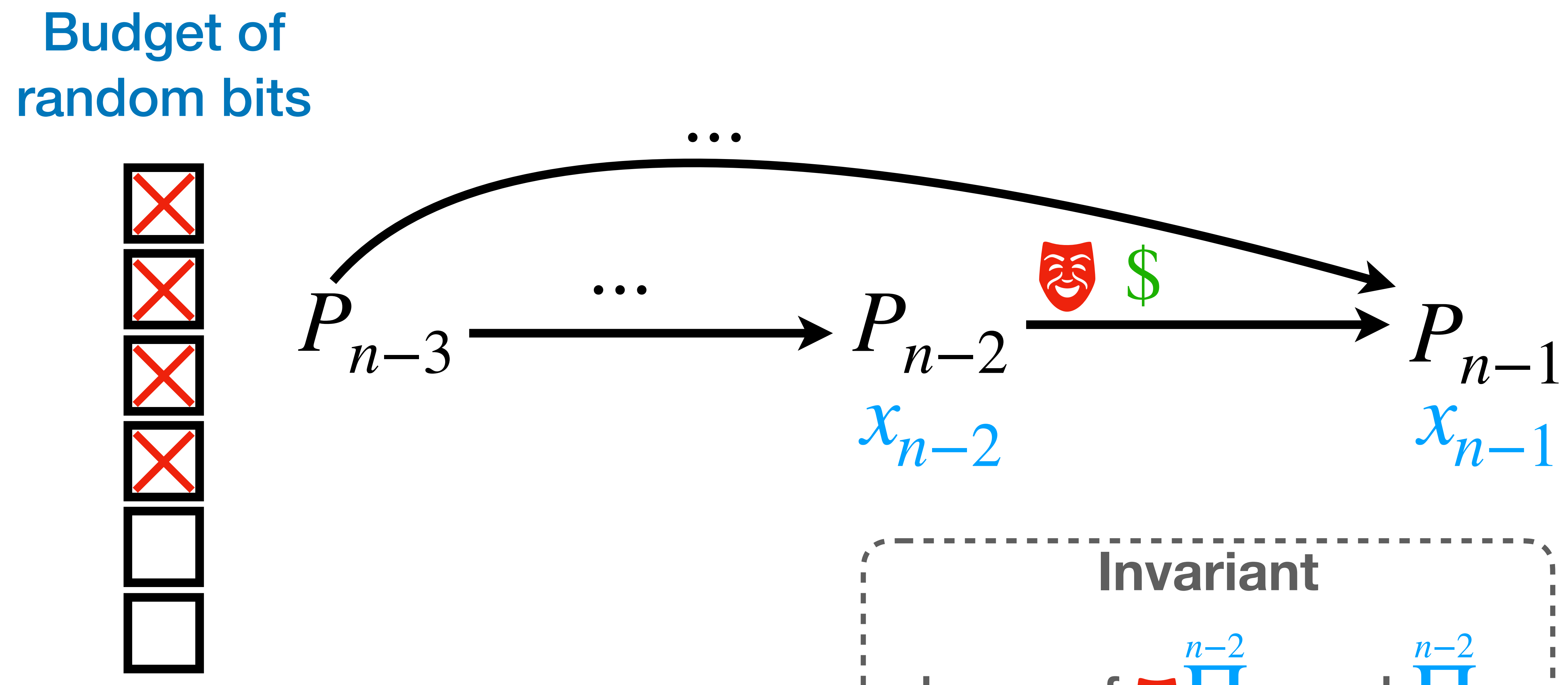
shares of $\prod_{i=0}^{n-2} x_i$ and $\prod_{i=0}^{n-2} x_i$

Idea 1

Use an oblivious transfer with x_{n-1} as selection bit, and sender inputs 0 and P_{n-2}'s share of $\prod_{i=0}^{n-1} x_i \implies$ uses 3 random bits!

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Output Phase



Invariant

shares of $\prod_{i=0}^{n-2} x_i$ and $\prod_{i=0}^{n-2} x_i$

Output

$$\begin{cases} 0 & \text{if } x_{n-1} = 0 \\ \prod_{i=0}^{n-1} x_i & \text{if } x_{n-1} = 1 \end{cases}$$

Idea 1

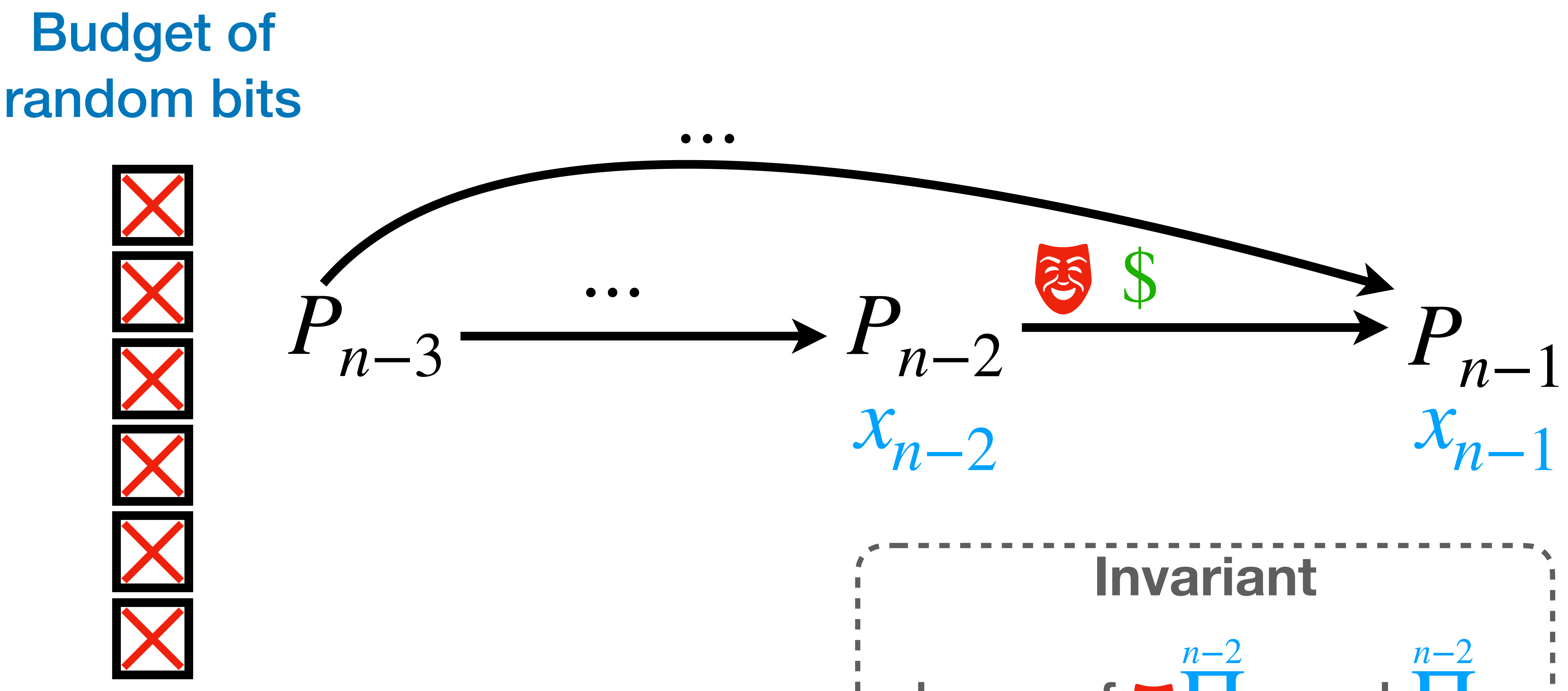
Use an oblivious transfer with x_{n-1} as selection bit, and sender inputs 0 and P_{n-2}'s share of $\prod_{i=0}^{n-1} x_i \Longrightarrow$ uses 3 random bits!

Idea 2

P_{n-2} and P_{n-1} don't need the rerandomization bit $\$ \Longrightarrow$ can reuse it for the OT!

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Output Phase



Invariant

shares of 🎭$\prod_{i=0}^{n-2} x_i$ and $\prod_{i=0}^{n-2} x_i$

Output

$$\begin{cases} 0 & \text{if } x_{n-1} = 0 \\ \prod_{i=0}^{n-1} x_i & \text{if } x_{n-1} = 1 \end{cases}$$

Idea 1

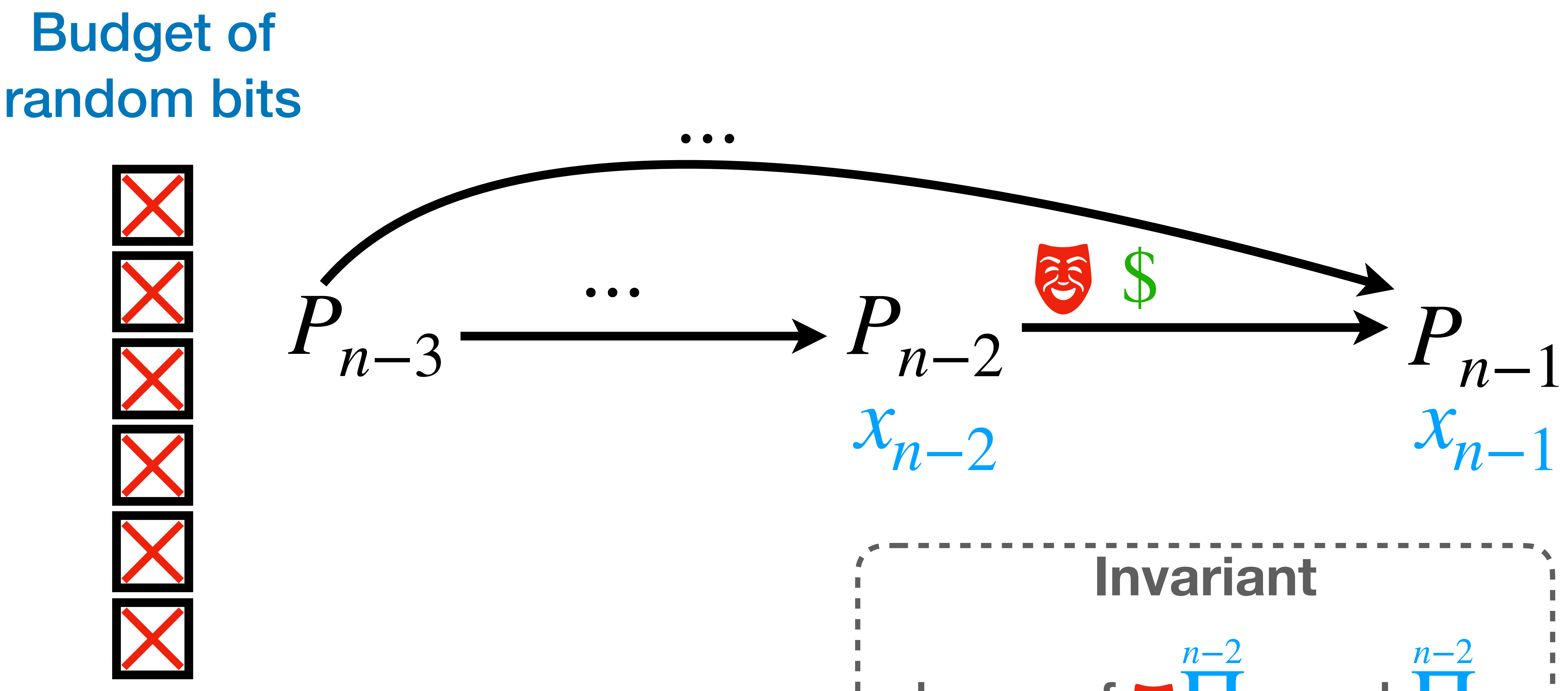
Use an oblivious transfer with x_{n-1} as selection bit, and sender inputs 0 and P_{n-2}'s share of $\prod_{i=0}^{n-1} x_i \Longrightarrow$ uses 3 random bits!

Idea 2

P_{n-2} and P_{n-1} don't need the rerandomization bit $\$ \Longrightarrow$ can reuse it for the OT!

Second Result: 1-Private n-Party AND with 6 Bits and 1 Source

Output Phase



Invariant

shares of $\prod_{i=0}^{n-2} x_i$ and $\prod_{i=0}^{n-2} x_i$

Output

$$\begin{cases} 0 & \text{if } x_{n-1} = 0 \\ \prod_{i=0}^{n-1} x_i & \text{if } x_{n-1} = 1 \end{cases}$$

Idea 1

Use an oblivious transfer with x_{n-1} as selection bit, and sender inputs 0 and P_{n-2}'s share of $\prod_{i=0}^{n-1} x_i \implies$ uses 3 random bits!

Idea 2

P_{n-2} and P_{n-1} don't need the rerandomization bit $\$ \implies$ can reuse it for the OT!

Thank you for your attention!

Questions?