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
Information-Theoretic Secure Computation

The Model

• $n$ 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

• parties with inputs $(x_1, \cdots, x_n)$
The Model

Information-Theoretic Secure Computation

- $n$ parties with inputs $(x_1, \ldots, x_n)$
- The adversary corrupts at most $t$ parties
- **Goal**: computing $f(x_1, \ldots, x_n)$ 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

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

How much money to compute $f(x_1, \ldots, x_n)$?
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: \textit{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 $\$\$\$\$\$\$ to compute $f(x_1, \cdots, x_n)$?
The Model

Information-Theoretic Secure Computation

- \( n \) parties with inputs \((x_1, \ldots, x_n)\)
- The adversary corrupts at most \( t \) parties
- **Goal:** computing \( f(x_1, \ldots, x_n) \) 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 \( f(x_1, \ldots, x_n) \)?
The Model

Information-Theoretic Secure Computation

- $n$ parties with inputs $(x_1, \ldots, x_n)$
- The adversary corrupts at most $t$ parties
- **Goal:** computing $f(x_1, \ldots, x_n)$ 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 $f(x_1, \ldots, x_n)$?
First Result: upper and lower bounds on the number of sources

<table>
<thead>
<tr>
<th>Deterministic Functionalities</th>
<th>Randomized Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>Upper Bound</td>
</tr>
<tr>
<td>$t$</td>
<td>$t + 1$</td>
</tr>
<tr>
<td>$t$</td>
<td>$t + 1$</td>
</tr>
</tbody>
</table>

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

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

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

This work: $t + 1$ sources suffice to $t$-privately compute any randomized functionality.
Kushilevitz & Mansour, PODC’96 show that $t$ parties must toss coins for $t$-private XOR

Previous work

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

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

<table>
<thead>
<tr>
<th>Deterministic Functionalities</th>
<th>Randomized Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>$t$</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>$t + 1$</td>
</tr>
</tbody>
</table>

$t$ parties

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

This work: there is a randomized functionality $ℱ$ that cannot be $t$-privately comp. with $t$ sources
**First Result: Random Source Complexity of $t$-Private Computation**

<table>
<thead>
<tr>
<th>Deterministic Functionalities</th>
<th>Randomized Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>Upper Bound</td>
</tr>
<tr>
<td>$t$</td>
<td>$t$</td>
</tr>
<tr>
<td>$t + 1$</td>
<td>$t + 1$</td>
</tr>
</tbody>
</table>

- **Kushilevitz & Mansour, PODC'96** show that $t$ parties must toss coins for $t$-private XOR.
- **Previous work**:
  - Lower Bound: $t$
  - Upper Bound: $t + 1$
- **First Result**: upper and lower bounds on the number of sources.
- **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.
- **This work**: there is a randomized functionality $\mathcal{F}$ that cannot be $t$-privately comp. with $t$ sources.

**Easy**
**First Result: Random Source Complexity of $t$-Private Computation**

<table>
<thead>
<tr>
<th>Deterministic Functionalities</th>
<th>Randomized Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound</td>
<td>$t$</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>$t$</td>
</tr>
</tbody>
</table>

**Core result**

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

**Previous work**

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**

$\begin{array}{cc}
    t & t + 1 \\
    t & t + 1 \\
\end{array}$

**Easy**

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

**This work**:

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.

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

Previous work

Core result

This work: It suffices that \( t \) parties toss coins to \( t \)-privately compute all deterministic functionalities.

Core result

This work: \( t + 1 \) sources suffice to \( t \)-privately compute any randomized functionality.

Deterministic Functionalities

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( t )</td>
<td>( t )</td>
</tr>
</tbody>
</table>

Randomized Functionalities

<table>
<thead>
<tr>
<th></th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>( t + 1 )</td>
<td>( t + 1 )</td>
</tr>
</tbody>
</table>

Secure computation is impossible deterministically... But it is possible even if we allow the adversary to corrupt all players that toss coins!
Second Result: Randomness vs Random Sources, and 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.
Kushilevitz & Mansour, PODC'96 show that $t$ parties must toss coins for $t$-private XOR.

Previous 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.

Core result:

<table>
<thead>
<tr>
<th>Deterministic Functionalities</th>
<th>Randomized Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$</td>
<td>$t+1$</td>
</tr>
<tr>
<td>$t$</td>
<td>$t+1$</td>
</tr>
</tbody>
</table>

Easy:

This work: there is a randomized functionality $F$ that cannot be $t$-privately comp. with $t$ sources.
<table>
<thead>
<tr>
<th>Deterministic Functionalities</th>
<th>Randomized Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower Bound: $t$</td>
<td>Upper Bound: $t + 1$</td>
</tr>
<tr>
<td>This work: It suffices that $t$ parties toss coins to $t$-privately compute all deterministic functionalities</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

Easy Core result.

First Result: Random Source Complexity of $t$-Private Computation

Starting point (warmup)
Kushilevitz & Mansour, PODC'96, show that $t$ parties must toss coins for $t$-private XOR.

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

**Deterministic Functionalities**
- Lower Bound: $t$
- Upper Bound: $t$

**Randomized Functionalities**
- Lower Bound: $t$
- Upper Bound: $t + 1$

This work: there is a randomized functionality $F$ that cannot be $t$-privately computed with $t$ sources.

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

Most of the presentation is dedicated to understanding the relationships between deterministic and randomized functionalities.
Kushilevitz & Mansour, PODC’96 show that $t$ parties must toss coins for $t$-private XOR.

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

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

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

First Result: Random Source Complexity of $t$-Private Computation

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

Easy - see paper!
Warmup: \((t + 1)\) sources, randomized functionalities

\[ F(x_1, x_2, x_3, x_4, x_5; r) \]
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

\[
F(x_1, x_2, x_3, x_4, x_5; r)
\]
All parties run BGW with these tapes to compute $F(x_1, x_2, x_3, x_4, x_5; r)$

**BGW** (Ben-Or, Goldwasser, Wigderson 1988): information-theoretic secure computation for any $t < n/2$. Any other IT protocol would work.
Warmup: \((t + 1)\) sources, randomized functionalities

All parties run BGW with these tapes to compute \(F(x_1, x_2, x_3, x_4, x_5; r)\)

**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

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

Core Result: $t$ sources, deterministic functionalities

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

$$F'(\text{shares, } x_2, x_3, x_5) = F(x_1, x_2, x_3, x_4, x_5)$$
Core Result: $t$ sources, deterministic functionalities

Outer Protocol

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
Core Result: $t$ sources, deterministic functionalities

Inner Protocol

Tape sharing
Core Result: $t$ sources, deterministic functionalities
Core Result: $t$ sources, deterministic functionalities

**Inner Protocol**

$n$-party BGW

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

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
Summary of the protocol

Core Result: $t$ sources, deterministic functionalities

Security

Two cases

(1) there is one honest source

(2) $𝒜$ corrupts only the sources
Summary of the protocol

**Security**

**Two cases**
1. There is one honest source
2. \( \mathcal{A} \) corrupts only the sources

**Two observations**
(a) \( \mathcal{A} \) can’t see the shared tapes
(b) \( t < n/2 \) \( \Rightarrow \) \( t < n - t \)

**Core Result:** \( t \) sources, deterministic functionalities
Summary of the protocol

Security

Two cases

1. Two observations
   - BGW is secure
   - The Beaver triples are trusted

   (a) $\mathcal{A}$ can’t see the shared tapes
   - $t < n/2$ implies $t < n - t$
   - there is one honest player
   - GMW is secure

2. One observation
   - no source ever sees an input-dependent message, beyond the output!
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: $\land_{i=0}^{n-1} x_i$
- 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

$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \ldots$

$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow x_3$
Second Result: 1-Private \( n \)-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \ldots$

Mask

$\cdot x_0 \rightarrow x_0$

Shares

$\times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \times \time...
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Mask

$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \ldots$

Shares

$x_0 \cdot x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$

$\cdot x_0 \quad x_0$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Invariants
- shares of $\prod_i x_i$ and $\prod_i x_i$

Shares

Mask

$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \ldots$

$x_0 \cdot x_0$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Mask

$P_0$ $P_1$ $P_2$ $P_3$ ...

$x_0$ $x_1$ $x_2$ $x_3$

Shares

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

Mask

 Shares

Invariant

shares of $\prod x_i$ and $\prod x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Mask

$x_0 \cdot \cdot \cdot$

Shares

Invariant

shares of $\prod_{i} x_i$ and $\prod_{i} x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Invariant shares of $\prod_i x_i$ and $\prod_i x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Invariant shares of $\prod_{i} x_i$ and $\prod_{i} x_i$

Invariant

shares of $\prod_{i} x_i$ and $\prod_{i} x_i$

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

Budget of random bits

Mask

$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \ldots$

Rerandomizer

$x_0 \cdot x_1 \rightarrow x_0 \cdot x_1$

Shares

Invariant
shares of $\prod_{i} x_i$ and $\prod_{i} x_i$

Invariant
shares of $\prod_{i} x_i$ and $\prod_{i} x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Invariants:
- shares of $\prod_i x_i$ and $\prod_i x_i$

Diagram:
- $P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \ldots$
- $x_0, x_1, x_2, x_3$
- Shares, Rerandomizer, Mask
- $x_0 \cdot x_1 \rightarrow x_0 \cdot x_1$
- $\prod_i x_i$ and $\prod_i x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

Mask

Rerandomizer

$x_0 \cdot x_0$

Shares

Invariant

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

Invariant

shares of $\prod_i x_i$ and $\prod_i x_i$
Second Result: 1–Private $n$–Party AND with 6 Bits and 1 Source

Main Phase

Budget of random bits

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

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

Invariant shares of $\prod_i x_i$ and $\prod_i x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Output Phase

Budget of random bits

Invariant shares of $\prod_{i=0}^{n-2} x_i$ and $\prod_{i=0}^{n-2} x_i$
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Output Phase

Budget of random bits

Invariant

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

Output

\[ \begin{align*}
0 & \quad \text{if } x_{n-1} = 0 \\
\prod_{i=0}^{n-1} x_i & \quad \text{if } x_{n-1} = 1
\end{align*} \]
Output Phase

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

Budget of random bits

\[ \prod_{i=0}^{n-1} x_i \]

Output

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

\[ \prod_{i=0}^{n-2} x_i \]

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

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

Idea 2
$P_{n-2}$ and $P_{n-1}$ don’t need the rerandomization bit $\$ \rightarrow \text{can reuse it for the OT}!
Second Result: 1-Private $n$-Party AND with 6 Bits and 1 Source

Output Phase

Budget of random bits

\[ P_{n-3} \rightarrow \ldots \rightarrow P_{n-2} \rightarrow P_{n-1} \]

\[ x_{n-2} \rightarrow x_{n-1} \]

Invariant shares of \( \prod_{i=0}^{n-2} x_i \) and \( \prod_{i=0}^{n-2} x_i \)

Output

\[ \begin{align*}
0 & \text{ if } x_{n-1} = 0 \\
\prod_{i=0}^{n-1} x_i & \text{ if } x_{n-1} = 1
\end{align*} \]

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!
Output Phase

Budget of random bits

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

Idea 2
$P_{n-2}$ and $P_{n-1}$ don’t need the rerandomization bit $\$ \Rightarrow$ can reuse it for the OT!
Thank you for your attention!

Questions?