Blockchain and secure computation

Vassilis Zikas
RPI

Winter School on Cryptocurrency and Blockchain Technologies
Shanghai Jiao Tong University
2017
| What is bitcoin and how does it work? | ✓ |
## Bitcoin

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is bitcoin and how does it work?</td>
<td>✔️</td>
</tr>
<tr>
<td>Is it secure?</td>
<td>(in restricted models)</td>
</tr>
</tbody>
</table>

(✔️) means yes, (❓) means no, (⭕) means undecided.
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is bitcoin and how does it work?</td>
<td>✓</td>
</tr>
<tr>
<td>Is it secure?</td>
<td>✓ ? (in restricted models)</td>
</tr>
<tr>
<td>What do we get from it?</td>
<td></td>
</tr>
<tr>
<td>What is bitcoin and how does it work?</td>
<td>✓</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>Is it secure?</td>
<td>✓ ?</td>
</tr>
<tr>
<td>What do we get from it?</td>
<td>(in restricted models)</td>
</tr>
</tbody>
</table>
What Crypto can get from Bitcoin?
What Crypto can get from Bitcoin?

In this talk

“Bitcoin = Ledger-based cryptocurrency”
What Crypto can get from Bitcoin?

In this talk
“Bitcoin = Ledger-based cryptocurrency”

A public transaction ledger

Use what is on this ledger
What Crypto can get from Bitcoin?

In this talk

“Bitcoin = Ledger-based cryptocurrency”

A public transaction ledger

A bulletin board with a filter on what gets written there

Use what is on this ledger
What Crypto can get from Bitcoin?

In this talk
“Bitcoin = Ledger-based cryptocurrency”

A public transaction ledger

A bulletin board with a filter on what gets written there

Use what is on this ledger
What is exactly the problem that bitcoin solves?
The Public Transaction Ledger

What is exactly the problem that bitcoin solves?

The core security goal of Bitcoin is to ensure that all parties establish a common and irreversible view of the sequence of transactions.
The Public Transaction Ledger

What is *exactly* the problem that bitcoin solves?

“Backbone” [GarayKiayiasLeonardos15, PassSeemanShelat16]

The core security goal of Bitcoin is to ensure that all parties establish a *common and irreversible view* of the sequence of transactions
The Public Transaction Ledger

What is exactly the problem that bitcoin solves?

“Backbone” [GarayKiayiasLeonardos15, PassSeemanShelat16]

The core security goal of Bitcoin is to ensure that all parties establish a common and irreversible view of the sequence of transactions

This goal can be captured as an ideal Transaction-Ledger Functionality
The Public Transaction Ledger

What is exactly the problem that bitcoin solves?

“Backbone” [GarayKiayiasLeonardos15, PassSeemanShelat16]

The core security goal of Bitcoin is to ensure that all parties establish a common and irreversible view of the sequence of transactions.

This goal can be captured as an ideal Transaction-Ledger Functionality.

A trusted third party that gives whomever accesses it the same power as using the Bitcoin network.
The Public Transaction Ledger

What is exactly the problem that Bitcoin solves?

“A trusted third party that gives whomever accesses it the same power as using the Bitcoin network”

“Backbone” [GarayKiayiasLeonardos15, PassSeemanShelat16]

The core security goal of Bitcoin is to ensure that all parties establish a common and irreversible view of the sequence of transactions.

This goal can be captured as an ideal Transaction-Ledger Functionality.

In this talk

“Bitcoin = Ledger-based cryptocurrency”
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network

\[ G_{\text{net}} \]
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network

\[ G_{\text{net}} \]
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network

\[ G_{net} \]
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network
The Public Transaction Ledger

A resource as an ideal functionality: Example. Communication network

Upon receiving $(i \rightarrow j, m)$ from Computer $i$ send $m$ to Computer $j$
The Public Transaction Ledger

A resource as an ideal functionality: The Bitcoin network
The Public Transaction Ledger

A resource as an ideal functionality: The Bitcoin network

Upon receiving \((i \rightarrow j, m)\) from Computer \(i\), send \(m\) to Computer \(j\)
The Public Transaction Ledger

A resource as an ideal functionality: The Bitcoin network

Upon receiving \((i \rightarrow j, m)\) from Computer \(i\) send \(m\) to Computer \(j\)
The Public Transaction Ledger

A resource as an ideal functionality: The Bitcoin network

Upon receiving \((i \rightarrow j, m)\) from Computer \(i\) send \(m\) to Computer \(j\)

\(G_{\text{net}}\)

\(G_{\text{ledger}}\)
The Public Transaction Ledger [KZZ16]
The Public Transaction Ledger [KZZ16]

GetState \rightarrow \text{"State"} \rightarrow \text{State}

\text{State} \rightarrow \text{St}

\mathcal{G}_{\text{ledger}}
The Public Transaction Ledger [KZZ16]
The Public Transaction Ledger [KZZ16]

(State, Stllx)

GetState

“State”

G
ledger

(Submit, x)
The Public Transaction Ledger [KZZ16]

- In reality: Not a Bulletin Board
- Inputs (transactions) are filtered
In reality: Not a Bulletin Board

Inputs (transactions) are filtered
The Public Transaction Ledger [KZZ16]

- In reality: Not a Bulletin Board
- Inputs (transactions) are filtered
In reality: Not a Bulletin Board
• Inputs (transactions) are filtered
The Public Transaction Ledger [KZZ16]

- In reality: Not a Bulletin Board
- Inputs (transactions) are filtered
The Public Transaction Ledger [KZZ16]

- In reality: Not a Bulletin Board
- Inputs (transactions) are filtered
The Public Transaction Ledger [KZZ16]

- In reality: Not a Bulletin Board
- Inputs (transactions) are filtered
The Public Transaction Ledger [KZZ16]

- In reality: Not a Bulletin Board
- Inputs (transactions) are filtered
- The order in which transactions in “State” are inserted might be adversarial … but not too adversarial
The Public Transaction Ledger [KZZ16]

Can reorder **the recently inserted** transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions.
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

GetState

State → Buffer → Validate(.) → (Submit, x)

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

- GetState
- Validate(.,)
- Buffer: $x_1, x_2, \ldots = \pi(x_1, \ldots)$
- Validate(.,)
  - Yes
  - No

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

- **State**
  - $x_1, x_2, \ldots = \pi(x_1, \ldots)$
  - GetState

- **Buffer**
  - Validate$(\cdot)$
  - (Submit, $x$)
  - (Permuted, $\pi$)

- **Validate**
  - Yes
  - No

- Can reorder **the recently inserted** transactions
Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

- **GetState**
- **State**
- **Buffer**
- **Validate(.)**
- **Blockify(.)**

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

Can reorder the recently inserted transactions
The Public Transaction Ledger [KZZ16]

More adversarial interference to have an accurate abstraction [BadetscherMaurerTschudiZikas17] (Also a construction from the Bitcoin network/protocol)

Can reorder the recently inserted transactions
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

Use what is on this ledger
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

Use what is on this ledger

How can we use it?
A simple e-voting protocol
A simple e-voting protocol

Tools 1/2: Threshold Encryption

• n-servers $S_1, \ldots, S_n$
• Each $S_i$ has secret key (share) $sk_i$
• There is one public key $pk$
A simple e-voting protocol

Tools 1/2: Threshold Encryption

- n-servers $S_1, \ldots, S_n$
- Each $S_i$ has secret key (share) $sk_i$
- There is one public key $pk$

- **Encryption**: Everyone with $pk$ can compute an encryption of message $m$, i.e., $c = \text{Enc}_{pk}(m)$
- **Decryption**: All $n$ servers together can decrypt, i.e., $\text{Dec}_{sk_1, \ldots, sk_n}(c) = m$
- **Threshold**: No $n-1$ servers can learn any information from the encryption
A simple e-voting protocol

Tool 2/2: Additive Homomorphic Encryption

Given ciphertexts $c_1 = \text{Enc}_{pk}(m_1)$ and $c_2 = \text{Enc}_{pk}(m_2)$
we can compute encryption $\text{Enc}_{pk}(m_1 + m_2)$
A simple e-voting protocol

Tool 2/2: Additive Homomorphic Encryption

Same encryption key

Given ciphertexts $c_1 = \text{Enc}_{pk}(m_1)$ and $c_2 = \text{Enc}_{pk}(m_2)$

we can compute encryption $\text{Enc}_{pk}(m_1 + m_2)$
A simple e-voting protocol

Setup

• n electoral authorities $S_1, \ldots, S_n$ with key shares $sk_1, \ldots, sk_n$ and pk.

To vote

• Each voter $V_i$ encrypts his vote $i_i (0 \text{ or } 1)$ and submits $c_i = Enc_{pk}(vote_i)$ to the BB.
• The votes are homomorphically tallied (i.e., $c = Enc_{pk}(vote_1 + vote_2 + \ldots)$)
• $c$ is decrypted by the electoral authorities.
A simple e-voting protocol

**Setup**
- $n$ electoral authorities $S_1, \ldots, S_n$ with key shares $sk_1, \ldots, sk_n$ and $pk$.

**To vote**
- Each voter $V_i$ encrypts his $vote_i$ (0 or 1) and submits $c_i = Enc_{pk}(vote_i)$ to the BB.
- The votes are homomorphically tallied (i.e., $c := Enc_{pk}(vote_1 + vote_2 + \ldots)$).
- $c$ is decrypted by the electoral authorities.
A simple e-voting protocol

Setup
- n electoral authorities $S_1,\ldots,S_n$ with key shares $sk_1,\ldots,sk_n$ and $pk$.

To vote
- Each voter $V_i$ encrypts his $vote_i$ (0 or 1) and submits $c_i=Enc_{pk}(vote_i)$ to the BB.
- The votes are homomorphically tallied (i.e., $c:=Enc_{pk}(vote_1 + vote_2 + \ldots)$).
- $c$ is decrypted by the electoral authorities.

Having a public transaction ledger ensures that
- The Bulletin Board where the votes are kept is decentralized, i.e., no server needs to be trusted to maintain it.
- The parties can see when the votes are added (no reordering is allowed).
- A vote that is added cannot be deleted.
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

What is on this ledger?
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

What is on this ledger?
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

What is on this ledger?

Random Stuff
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

What is on this ledger?

Random Stuff

Money
What Crypto can get from Bitcoin?

A public transaction ledger
A bulletin board with a filter on what gets written there

What is on this ledger?
- Random Stuff
- Money
- ??
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

Use what is on this ledger

Random Stuff

Money

??
The Bitcoin ledger as a random beacon

\( (t_1, 0110), (t_2, 0001) \ldots \)
The Bitcoin ledger as a random beacon

Why is this useful?

(t₁,0110), (t₂,0001) …
The Bitcoin ledger as a random beacon

Why is this useful?

- Lotteries:

\((t_1, 0110), (t_2, 0001) \ldots\)
The Bitcoin ledger as a random beacon

Why is this useful?

- Lotteries:

$(t_1, 0110), (t_2, 0001)$ ...
The Bitcoin ledger as a random beacon

(t₁,0110), (t₂,0001) …

Why is this useful?

• Lotteries:
  • Before time t₂: collect tokens x₀₀₀₀, x₀₀₀₁,…
The Bitcoin ledger as a random beacon

Why is this useful?

• Lotteries:
  • Before time $t_2$: collect tokens $x_{0000}$, $x_{0001}$, ...
  • At time $t_2$: The token indexed by the beacon’s value wins
The Bitcoin ledger as a random beacon

Why is this useful?

Lotteries:

• Before time $t_2$: collect tokens $x_{0000}$, $x_{0001}$, ...
• At time $t_2$: The token indexed by the beacon’s value wins
The Bitcoin ledger as a random beacon

Why is this useful?

- Lotteries:
  - Before time $t_2$: collect tokens $x_{0000}$, $x_{0001}$, ...
  - At time $t_2$: The token indexed by the beacon’s value wins

- Zero-knowledge Proofs

- Common Random String (aka the cryptographer’s paradise)
The Bitcoin ledger as a random beacon

\[(t_1, 0110), (t_2, 0001) \ldots\]

**Why is this useful?**

**Is it possible?**

- **Heuristically**: Hash each block [AndrychowiczDziembowski15]
- **No**: if we require the rate of the beacon to be the same as the Bitcoin network [BentovGabizonKiayiasZhouZikasZuckerman17]
- **Yes**: if we allow a much slower beacon rate
  - Under number theoretic assumptions [LenstraWesolowski15]
  - Assuming (only) random oracles [ongoing …]
What Crypto can get from Bitcoin?

A public transaction ledger

A bulletin board with a filter on what gets written there

Use what is on this ledger

Random Stuff

Money

??
What Crypto can get from Bitcoin?

A public transaction ledger
A bulletin board with a filter on what gets written there

Use what is on this ledger

Random Stuff

Money

People (good or bad) want money
What Crypto can get from Bitcoin?

A public transaction ledger
A bulletin board with a filter on what gets written there

Use what is on this ledger

Random Stuff

Money

People (good or bad) want money

We can use bitcoins as compensation for relaxed security
Leveraging Security Loss with Coins

... in Secure Multi-Party Computation (MPC)
Leveraging Security Loss with Coins

... in Secure Multi-Party Computation (MPC)
Multi-Party Computation (MPC)

**Goal:** Parties $P_1, \ldots, P_n$ with inputs $x_1, \ldots, x_n$ wish to compute a function $f(x_1, \ldots, x_n)$ securely.
Multi-Party Computation (MPC)

Ideal World

\[ f(x_1, x_2, \ldots, x_n) = y \]
Multi-Party Computation (MPC)

Ideal World

Real World
Multi-Party Computation (MPC)

Ideal World

\[ f^{f}(x_1, x_2, \ldots, x_n) = y \]

Real World

\[ \pi_1(x_1) \leftrightarrow \pi_2(x_2) \leftrightarrow \cdots \leftrightarrow \pi_n(x_n) \]
Multi-Party Computation (MPC)

Ideal World

\[ \mathcal{F}^f \]

\[ \begin{array}{c}
   x_1 \\
   P_1
\end{array} \xrightarrow{f(x)} \begin{array}{c}
   x_2 \\
   f(\bar{x}) \\
   \downarrow \\
   P_2 \\
   \vdots \\
   x_n \\
   f(\bar{x})=y
\end{array} \]

Real World

\[ \begin{array}{c}
   \pi_1(x_1) \\
   P_1
\end{array} \leftrightarrow \begin{array}{c}
   \pi_2(x_2) \\
   P_2 \\
   \vdots \\
   \pi_n(x_n)
\end{array} \]

\[ \ll \]
Multi-Party Computation (MPC)

Ideal World

\[ F^f \]

\[ \begin{align*}
  x_1 &\xrightarrow{f(x)} P_1 \\
  x_2 &\xrightarrow{f(x)} P_2 \\
  \vdots \\
  x_n &\xrightarrow{f(x)} y
\end{align*} \]

Protocol \( \pi \) is secure if for every adversary:

- **(privacy)** Whatever the adversary learns he could compute by himself
- **(correctness)** Honest (uncorrupted) parties learn their correct outputs

Real World

\[ \begin{align*}
  \pi_1(x_1) &\leftrightarrow P_1 \\
  \pi_2(x_2) &\leftrightarrow P_2 \\
  \vdots \\
  \pi_n(x_n) &\leftrightarrow P_n
\end{align*} \]
Multi-Party Computation (MPC)

Ideal World

\[ f(x) = y \]

Real World

Protocol \( \pi \) is secure if for every adversary:

- (privacy) Whatever the adversary learns he could compute by himself
- (correctness) Honest (uncorrupted) parties learn their correct outputs

Private blockchains are a special case
In **fair MPC**: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output.
In fair MPC: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output.
**Fair MPC**

**In fair MPC:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output

![Diagram showing a function $f$ and two inputs $P_1$ and $P_2$]
In fair MPC: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output

Fair MPC is impossible against corrupted majorities
**Fair MPC**

**In fair MPC:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output.

Fair MPC is impossible against corrupted majorities.

Security against corrupted majorities = Security with abort

(Unfair)
Fair MPC

In fair MPC: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output.

Fair MPC is impossible against corrupted majorities.

Security against corrupted majorities \(=\) Security with abort

Discounted security
MPC with Fair Compensation

**Idea** [AndrychowiczDziembowskiMalinowskiMazurek14]:
We can leverage unfairness with $$$

**MPC with fair compensation:** If the adversary learns any information on the output beyond (what is derived by) its inputs then every honest party should learn the output **or** get compensated.
MPC with Fair Compensation

**Idea** [AndrychowiczDziembowskiMalinowskiMazurek14]: We can leverage unfairness with $$$

**MPC with fair compensation**: If the adversary learns any information on the output beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.

\[
F_f \Downarrow P_1 \Downarrow \Downarrow \Downarrow y \quad \text{(Unfair)}
\]
MPC with Fair Compensation

Idea [AndrychowiczDziembowskiMalinowskiMazurek14]: We can leverage unfairness with $$$

MPC with fair compensation: If the adversary learns any information on the output beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.

\[ F^f \]

- \( P_1 \)
- \( P_2 \)
- \( y \)

\[ \text{✘ (Unfair)} \]
MPC with Fair Compensation

**Idea** [AndrychowiczDziembowskiMalinowskiMazurek14]: We can leverage unfairness with $$$

**MPC with fair compensation**: If the adversary learns any information on the output beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.

![Diagram](image-url)
MPC with Fair Comp.: Construction

Tools 1/2: Authenticated Additive Secret Sharing

\[ x = x_1 \oplus \ldots \oplus x_n, \quad (sk, vk) \leftarrow \text{KeyGen} \]

- \[ [x]_1 = x_1, \operatorname{Sig}_{sk}(id_1, x_1), vk \]
- \[ \ldots \]
- \[ [x]_n = x_n, \operatorname{Sig}_{sk}(id_n, x_n), vk \]
MPC with Fair Comp.: Construction

[ BentovKumaresan14,15 ]

Tools 1/2: Authenticated Additive Secret Sharing

\[ x = x_1 \oplus \ldots \oplus x_n, (sk, vk) \leftarrow \text{KeyGen} \]

- \( P_1 \)
- \( \ldots \)
- \( P_n \)

\[ [x]_1 = x_1, \text{Sig}_{sk}(id_1, x_1), vk \]
\[ [x]_n = x_n, \text{Sig}_{sk}(id_n, x_n), vk \]

- No \( n-1 \) parties have info on \( x \)
- Together all \( n \) parties can recover \( x \)
- No party can lie about its share
  - Only \( x \) might be reconstructed!
MPC with Fair Comp.: Construction

Tools 2/2 : Claim and Refund Transactions

S transfers q coins to R such that
Tools 2/2 : Claim and Refund Transactions

S transfers $q$ coins to $R$ such that

- Time restriction $\tau$
MPC with Fair Comp.: Construction

[ BentovKumaresan14,15 ]

Tools 2/2 : Claim and Refund Transactions

S transfers q coins to R such that

- Time restriction \( \tau \)

<table>
<thead>
<tr>
<th>time</th>
<th></th>
</tr>
</thead>
</table>


Tools 2/2: Claim and Refund Transactions

S transfers q coins to R such that

- Time restriction $\tau$
S transfers q coins to R such that

- Time restriction $\tau$

A predicate (relation) $R(state, buffer, tx)$:
- In order to spend the coins the receiver needs to submit a tx satisfying $R$ (at the point of validation).
MPC with Fair Comp.: Construction

[ BentovKumaresan14,15 ]

Tools 2/2: Claim and Refund Transactions

S transfers q coins to R such that

- Time restriction $\tau$

- A predicate (relation) $R(state, buffer, tx)$:
  - In order to spend the coins the receiver needs to submit a tx satisfying $R$ (at the point of validation).

- Supported by Bitcoin scripting language
- Captured by $\text{Validate}(.)$
Protocol Idea for computing $y = f(x_1, \ldots, x_n)$

1. Run SFE with unfair abort to compute $\text{n-out-of-n}$ authenticated sharing $[y]$ of $y = f(x_1, \ldots, x_n)$
   - E.g., Every $P_i$ receives share $[y]_i$ such that $y = [y]_1 + \ldots + [y]_n$ and public signature on $[y]_i$
Protocol Idea for computing $y = f(x_1, \ldots, x_n)$

1. Run SFE with unfair abort to compute $n$-out-of-$n$ authenticated sharing $[y]$ of $y = f(x_1, \ldots, x_n)$
   
   - E.g., Every $P_i$ receives share $[y]_i$ such that $y = [y]_1 + \ldots + [y]_n$ and public signature on $[y]_i$
MPC with Fair Comp.: Construction

[Classic Works]

Protocol Idea for computing $y = f(x_1, \ldots, x_n)$

1. Run SFE with unfair abort to compute $n$-out-of-$n$ authenticated sharing $[y]$ of $y = f(x_1, \ldots, x_n)$
   - E.g., Every $P_i$ receives share $[y]_i$ such that $y = [y]_1 + \ldots + [y]_n$ and public signature on $[y]_i$

............................... Abort at this point is fair ...............................
2. Use the following reconstruction idea:

2.1. Every $P_i$ transfers 1 bitcoin to every $P_j$ with the restriction:
   
   - $P_j$ can claim (spend) this coin if it submits to the ledger his valid share (and signature) by round $\rho_{ij}$
   - if $P_j$ has not claimed this coin by the end of round $\rho_{ij}$, then the coin is “refunded” to $P_i$ (i.e., after round $\rho_{ij}$, $P_i$ can spend this coin himself).
MPC with Fair Comp.: Construction

[ BentovKumaresan14,15]

Protocol Idea for computing $y=f(x_1,...,x_n)$

2. Use the following reconstruction idea:

2.1. Every $P_i$ transfers 1 bitcoin to every $P_j$ with the restriction:

- $P_j$ can claim (spend) this coin if it submits to the ledger his valid share (and signature) by round $\rho_{ij}$
- if $P_j$ has not claimed this coin by the end of round $\rho_{ij}$, then the coin is “refunded” to $P_i$ (i.e., after round $\rho_{ij}$, $P_i$ can spend this coin himself).

2.2. Proceed in rounds in which the parties claim the coins from other parties by announcing their shares (and signatures)
MPC with Fair Comp.: Construction

Protocol Idea for computing $y=f(x_1,\ldots,x_n)$

Security (SFE with fair compensation): Follow the money …

- If the adversary announces all his shares then every party:
  - Sends $n$ coins in phase two (one to each party)
  - Claims back $n$ coins in phase three (one from each party)

- If a corrupted party $P_j$ does not announce his share then every party
  - Sends $n$ coins in phase two (one to each party)
  - Claims back
    - $n$ coins in phase three for announcing his shares
    - the coin that it had sent to $P_j$
Rethinking MPC with Fair Compensation

[Reference: BentovKumaresan14,15]
Rethinking MPC with Fair Compensation

Time

-------------------- Protocol Starts
Rethinking MPC with Fair Compensation

[ BentovKumaresan14,15 ]

Time

<table>
<thead>
<tr>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Starts</td>
</tr>
<tr>
<td>Sharing is Output, Committed transactions</td>
</tr>
</tbody>
</table>
Rethinking MPC with Fair Compensation

[BentovKumaresan14,15]

- Protocol Starts
- Sharing is Output, Committed transactions
- 1 hour: Start reclaiming transactions
Rethinking MPC with Fair Compensation

[ BentovKumaresan14,15 ]

Time

- Seconds: Protocol Starts, Sharing is Output, Committed transactions
- 1 hour: Start reclaiming transactions
- several hours: Output or compensation is settled
Rethinking MPC with Fair Compensation

[BentovKumaresan14,15]

Time

- Protocol Starts
- Sharing is Output, Committed transactions
- Start reclaiming transactions
- several hours output or compensation is settled

“several” =
- [BentovKumaresan14] linear in players (n)
- [BentovKumaresan15] constant
Rethinking MPC with Fair Compensation

What if the adversary aborts before making the committed transactions?

Time

- Protocol Starts
- Sharing is Output, Committed transactions
- Start reclaiming transactions
- several
- output or compensation is settled

“several” =
- [BentovKumaresan14] linear in players (n)
- [BentovKumaresan15] constant
Rethinking MPC with Fair Compensation

What if the adversary aborts before making the committed transactions?

Protocol Starts

Seconds

Sharing is Output, Committed transactions

1 hour

Start reclaiming transactions

several hours

output or compensation is settled

“several” =
  • [BentovKumaresan14] linear in players (n)
  • [BentovKumaresan15] constant
Rethinking MPC with Fair Compensation

What if the adversary aborts before making the committed transactions?

Protocol Starts

Seconds

Sharing is Output, Committed transactions

1 hour

Start reclaiming transactions

several hours

output or compensation is settled

“several” =
- [BentovKumaresan14] linear in players (n)
- [BentovKumaresan15] constant
Rethinking MPC with Fair Compensation

What if the adversary aborts before making the committed transactions?

Protocol Starts

Seconds

Sharing is Output, Committed transactions

1 hour

Start reclaiming transactions

several hours

output or compensation is settled

“several” =

- [BentovKumaresan14] linear in players (n)
- [BentovKumaresan15] constant
Rethinking MPC with Fair Compensation

What if the adversary aborts before making the committed transactions?

This can be confirmed here …

… and reclaimed here …

Output or compensation is settled

“several” =
  • [BentovKumaresan14] linear in players (n)
  • [BentovKumaresan15] constant
**Rethinking MPC with Fair Compensation**

**SFE with fair compensation:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.

\[ F_f \]

\[ \begin{align*}
P_1 & \quad \perp \\
+ & \quad + \\
α & \quad α \\
\downarrow & \quad \downarrow \\
\smiley & \quad \smiley \\
\end{align*} \quad \begin{align*}
\perp & \quad \perp \\
+ & \quad + \\
α & \quad α \\
\downarrow & \quad \downarrow \\
\smiley & \quad \smiley \\
\end{align*} \quad \begin{align*}
y \quad \& \\
\downarrow & \quad \downarrow \\
\smiley & \quad \smiley \\
\downarrow & \quad \downarrow \\
\frown & \quad \frown \\
\end{align*} \]

- \( \times \) (Unfair)
- \( \checkmark \) (“fair”)

\( F_f \)
SFE with fair compensation: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.
SFE with fair compensation: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.
Rethinking MPC with Fair Compensation

**SFE with fair compensation:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.
MPC with Robust Compensation [KZZ16]
MPC with Robust Compensation [KZZ16]

**Fair MPC:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output
MPC with Robust Compensation [KZZ16]

**Fair MPC:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output.

**MPC with fair compensation:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated.
MPC with Robust Compensation [KZZ16]

Fair MPC: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output

MPC with fair compensation: If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated (fast ...)

robust
MPC with Robust Compensation [KZZ16]

**Fair MPC:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output.

**robust MPC with fair compensation:** If the adversary learns any information beyond (what is derived by) its inputs then every honest party should learn the output or get compensated (fast …)

**How can we get robustness?**
MPC with Robust Compen. : Construction

Tools 1/3 : Special Transaction

S transfers q coins to R such that
S transfers q coins to R such that
- Time restriction $(\tau_-, \tau_+)$
S transfers q coins to R such that

- Time restriction \((\tau_-, \tau_+)\)

\[
\text{time}
\]
MPC with Robust Compen. : Construction

Tools 1/3 : Special Transaction

S transfers q coins to R such that

- Time restriction $(\tau_-, \tau_+)$

<table>
<thead>
<tr>
<th>Time</th>
<th>$\tau_-$</th>
<th>$\tau_+$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coins</td>
<td>blocked</td>
<td>R can claim coins</td>
</tr>
</tbody>
</table>
MPC with Robust Compen. : Construction

Tools 1/3 : Special Transaction

S transfers q coins to R such that

- Time restriction \((\tau_-, \tau_+)\)

- Link: A reference \(\text{ref}\) such that only a transaction with the same reference can spend the q coins
MPC with Robust Compen. : Construction

Tools 1/3 : Special Transaction

S transfers q coins to R such that

- **Time restriction** $(\tau_-, \tau_+)$

  - **Link**: A reference $\text{ref}$ such that only a transaction with the same reference can spend the q coins

  - **A predicate (relation)** $\mathcal{R}(\text{state}, \text{buffer}, \text{tx})$:
    - In order to spend the coins the receiver needs to submit a tx satisfying $\mathcal{R}$ (at the point of validation).
MPC with Robust Compen. : Construction

Tools 1/3 : Special Transaction

S transfers q coins to R such that

- Time restriction \((\tau_-, \tau_+)\)

- Link: A reference \(\text{ref}\) such that only a transaction with the same reference can spend the q coins

- A predicate (relation) \(R(\text{state}, \text{buffer}, \text{tx})\):
  - In order to spend the coins the receiver needs to submit a tx satisfying \(R\) (at the point of validation).

\[\mathcal{B}_{v, \text{address}_i, \text{address}_j, \Sigma, \text{aux}, \sigma_i, \tau}\]
MPC with Robust Compen. : Construction

Tools 1/3 : Special Transaction

S transfers q coins to R such that

- **Time restriction** \((\tau_-, \tau_+)\)

- **Link**: A reference \(\text{ref}\) such that only a transaction with the same reference can spend the q coins

- **A predicate (relation)** \(\mathcal{R}(\text{state}, \text{buffer}, \text{tx})\):
  - In order to spend the coins the receiver needs to submit a tx satisfying \(\mathcal{R}\) (at the point of validation).
MPC with Robust Compensation: Construction

Tools 2/3: Semi-honest MPC

An SFE protocol which is secure when parties follow their instructions.
**MPC with Robust Compen. : Construction**

**Tools 2/3 : Semi-honest MPC**

An MPC protocol which is secure when parties follow their instructions

**Example:** A Summation protocol

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$x_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_2$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$P_n$</td>
<td>$x_n$</td>
</tr>
</tbody>
</table>

$$y_1 + y_2 + \cdots + y_n = \sum_{i=1}^{n} y_i$$
MPC with Robust Compen. : Construction

Tools 2/3 : Semi-honest MPC

An MPC protocol which is secure when parties follow their instructions

**Example:** A Summation protocol

\[
\begin{array}{c|ccc}
 & P_1 & P_2 & P_n \\
\hline
P_1 & x_1 & x_{11} & x_{12} \cdots x_{1n} \\
P_2 & x_2 & & \\
\vdots & & & \\
P_n & x_n & & \\
\end{array}
\]

\[
x_1 = \bigoplus_{j=1}^{n} x_{1j}
\]
MPC with Robust Compen. : Construction

Tools 2/3 : Semi-honest MPC
An MPC protocol which is secure when parties follow their instructions

**Example:** A Summation protocol

<table>
<thead>
<tr>
<th></th>
<th>P₁</th>
<th>P₂</th>
<th>Pₙ</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>x₁</td>
<td>x₁₁</td>
<td>x₁₂</td>
</tr>
<tr>
<td>P₂</td>
<td>x₂</td>
<td>x₂₁</td>
<td>x₂₂</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pₙ</td>
<td>xₙ</td>
<td>xₙ₁</td>
<td>xₙ₂</td>
</tr>
</tbody>
</table>

\[
x₁ = \bigoplus_{j=1}^{n} x_{₁j}
\]

\[
x₂ = \bigoplus_{j=1}^{n} x_{₂j}
\]

\[
xₙ = \bigoplus_{j=1}^{n} x_{ₙj}
\]
MPC with Robust Compen. : Construction

Tools 2/3 : Semi-honest MPC

An MPC protocol which is secure when parties follow their instructions

Example: A Summation protocol

<table>
<thead>
<tr>
<th></th>
<th>P_1</th>
<th>P_2</th>
<th>P_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_1</td>
<td>x_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_2</td>
<td></td>
<td>x_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_1</td>
<td>y_2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{P}_1 & \quad x_1 & \quad x_{11} & \quad x_{12} & \cdots & \quad x_{1n} \\
\text{P}_2 & \quad x_2 & \quad x_{21} & \quad x_{22} & \cdots & \quad x_{2n} \\
\vdots & \quad \vdots & \quad \vdots & \quad \vdots & \quad \vdots & \quad \vdots \\
\text{P}_n & \quad x_n & \quad x_{n1} & \quad x_{n2} & \cdots & \quad x_{nn} \\
\end{align*}
\]

\[
\begin{align*}
x_1 &= \bigoplus_{j=1}^{n} x_{1j} \\
x_2 &= \bigoplus_{j=1}^{n} x_{2j} \\
x_n &= \bigoplus_{j=1}^{n} x_{nj} \\
y &= \bigoplus_{i=1}^{n} y_i
\end{align*}
\]
MPC with Robust Compen. : Construction

Tools 2/3 : Semi-honest MPC

An MPC protocol which is secure when parties follow their instructions

**Example:** A Summation protocol

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>$x_1$</td>
<td>$x_{11}$ $x_{12}$ $\cdots$ $x_{1n}$</td>
<td></td>
</tr>
<tr>
<td>$P_2$</td>
<td>$x_2$</td>
<td>$x_{21}$ $x_{22}$ $\cdots$ $x_{2n}$</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$P_n$</td>
<td>$x_n$</td>
<td>$x_{n1}$ $x_{n2}$ $\cdots$ $x_{nn}$</td>
<td></td>
</tr>
</tbody>
</table>

$x_1 = \bigoplus_{j=1}^{n} x_{1j}$

$x_2 = \bigoplus_{j=1}^{n} x_{2j}$

$x_n = \bigoplus_{j=1}^{n} x_{nj}$

$y_1 \ y_2 \ \cdots \ y_n$

$y = \bigoplus_{i=1}^{n} y_i$

Secure (private) against arbitrary many colluding parties.
MPC with Robust Compen. : Construction

Tools 2/3 : Semi-honest MPC
An MPC protocol which is secure when parties follow their instructions

Assuming a public key infrastructure (commitments/encryption/signatures) there exists a semi-honest MPC protocol \( \pi \) for every function which

- Uses only public communication
- Tolerates arbitrary many semi-honest parties
- Terminates in constant rounds
MPC with Robust Compen. : Construction

Tools 3/3 : The GMW Compiler
Compile a semi-honest MPC protocol $\pi$ into (malicious) secure
Compile a semi-honest MPC protocol \( \pi \) into (malicious) secure MPC with Robust Compen.

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Every \( P_i \) commits to its input

**Rounds 2 \ldots \rho_\pi + 1:**
Execute \( \pi \) round-by-round so that in each round every party proves (in ZK) that he follows \( \pi \)

**Tools 3/3: The GMW Compiler**
MPC with Robust Compen. : Construction

Tools 3/3 : The GMW Compiler

Compile a semi-honest MPC protocol $\pi$ into (malicious) secure

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Every $P_i$ commits to its input

**Rounds 2 ... $\rho_{\pi} + 1$:**
Execute $\pi$ round-by-round so that in each round every party proves (in ZK) that he follows $\pi$

---

**Security (with abort):**

- **Privacy:** The parties see the following:
  - Setup
  - Commitments
  - Messages from $\pi$

- **Correctness:**
  - If ZKPs succeed then the parties are indeed following $\pi$
  - Else abort
MPC with Robust Compen: Construction

Idea: Use “GMW”-like compiler on the Ledger
Idea: Use “GMW”-like compiler on the Ledger

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Every $P_i$ commits to its input

Rounds 2 ... $\rho_\pi + 1$:
Execute $\pi$ round-by-round so that in each round every party proves (in ZK) that he follows $\pi$
Idea: Use “GMW”-like compiler on the Ledger

**GMW**

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Every $P_i$ commits to its input

**Rounds 2 ... $\rho_\pi + 1$:**
Execute $\pi$ round-by-round so that in each round every party proves (in ZK) that he follows $\pi$

**GMW’:**

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Do nothing

**Round 2:**
Every $P_i$ commits to its input and broadcasts his view of the public setup.

**Rounds 3 ... $\rho_\pi + 2$:**
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that he follows $\pi$
Idea: Use “GMW”-like compiler on the Ledger

GMW’:

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every $P_i$ commits to its input and broadcasts his view of the public setup.

Rounds 3 … $\rho_\pi + 2$:
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows $\pi$
Idea: Use “GMW”-like compiler on the Ledger

\[ \text{GMW'}: \quad \text{MPC with Robust Compensation} \]

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Do nothing

**Round 2:**
Every \( P_i \) commits to its input and broadcasts his view of the public setup.

**Rounds 3 \ldots \rho_\pi + 2:**
Execute \( \pi \) round-by-round so that in each round every party proves (in NIZK) that the follows \( \pi \)
MPC with Robust Compen. : Construction

Idea: Use “GMW”-like compiler on the Ledger

GMW’: MPC with Robust Compensation

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every $P_i$ commits to its input and broadcasts his view of the public setup.

Rounds 3 ... $\rho + 2$: Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows $\pi$
MPC with Robust Compensation: Construction

Idea: Use “GMW”-like compiler on the Ledger

GMW':

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every $P_i$ commits to its input and broadcasts his view of the public setup.

Rounds 3 ... $\rho_\pi + 2$:
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows $\pi$
MPC with Robust Compen. : Construction

Idea: Use “GMW”-like compiler on the Ledger

GMW’:

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every P_i commits to its input and broadcasts his view of the public setup.

Rounds 3 ... \rho_\pi + 2:
Execute \pi round-by-round so that in each round every party proves (in NIZK) that the follows \pi

MPC with Robust Compensation

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Every party P_i makes n \cdot \rho_\pi special 1-coin transactions B_{(i,j,r)}:
- P_j can spend coin in round r
- ref needs to have the protocol ID
- R is true if the transaction which spends the coin includes a valid r-round message for P_j
Idea: Use “GMW”-like compiler on the Ledger

GMW’:
Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every $P_i$ commits to its input and broadcasts his view of the public setup.

Rounds 3 ... $\rho_\pi + 2$:
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows $\pi$

MPC with Robust Compensation

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Every party $P_i$ makes $n \cdot \rho_\pi$ special 1-coin transactions $B_{(i,j,r)}$:
- $P_j$ can spend coin in round $r$
- ref needs to have the protocol ID
- $R$ is true if the transaction which spends the coin includes a valid $r$-round message for $P_j$

Rounds 3 ... $\rho_\pi + 2$: Execute GMW($\pi$) round-by-round so that in each round $r$ every party spends all its round $r$ referenced coins by a transaction which includes the round $r$ message in GMW($\pi$).
MPC with Robust Compen.: Construction

Idea: Use “GMW”-like compiler on the Ledger

**GMW’**: 

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Do nothing

**Round 2:**
Every $P_i$ commits to its input and broadcasts his view of the public setup.

**Rounds 3 … $\rho_{\pi} + 2$:**
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows

MPC with Robust Compensation

**Round 0:**
Setup generation (+ commitments to randomness)

**Round 1:**
Every party $P_i$ makes $n \cdot \rho_{\pi}$ special 1-coin transactions $B_{(i,j,r)}$:
- $P_j$ can spend coin in round $r$
- ref needs to have the protocol ID
- $R$ is true if the transaction which spends the coin includes a valid $r$-round message for $P_j$

**Rounds 3 … $\rho_{\pi} + 2$:** Execute $\text{GMW}(\pi)$ round-by-round so that in each round $r$ every party spends all its round $r$ referenced coins by a transaction which includes the round $r$ message in $\text{GMW}(\pi)$.

$\text{Validate}(.)$ executes the code of an extra party without inputs in GMW and rejects if abort.
Security with Robust Compensation.

- **Case 1:** The adversary correctly makes all the "committing" transactions in Round 1
  
  - If no party cheats then every party claims from each of the other parties as many coins as he deposited by simply executing his protocol.
  
  - If some party $P_j$ cheats, then every party still claims all his coins as above + all the committed coins that $P_j$ cannot spend as he did not execute his protocol.
Security with Robust Compensation.

• **Case 2:** Some corrupted party does not make (consistent) transactions in Round 1
  
  • e.g. aborts or commits to a different setup.
Security with Robust Compensation.

- **Case 2:** Some corrupted party does not make (consistent) transactions in Round 1
  - e.g. aborts or commits to a different setup.

... seems to have similar issue as before ...
Security with Robust Compensation.

- **Case 2:** Some corrupted party does not make (consistent) transactions in Round 1
  - e.g. aborts or commits to a different setup.

  … seems to have similar issue as before …

- **Solution:** The validation predicate can be changed as:
  - Separates the parties into “islands” of consistent setups (depending on their Round-1 transactions).
  - For each island $I \subseteq [n]$: Compute the function among parties in $I$ (with all other parties’ input being 0)
MPC with Robust Compen. : Construction

Idea: Use “GMW”-like compiler on the Ledger

GMW’:

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every $P_i$ commits to its input and broadcasts his view of the public setup.

Rounds 3 … $\rho_{\pi} + 2$:
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows $\pi$

MPC with Robust Compensation

Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Every party $P_i$ makes $n \cdot \rho_{\pi}$ special 1-coin transactions $B_{(i,j,r)}$:
- $P_j$ can spend coin in round $r$
- ref needs to have the protocol ID
- $R$ is true if the transaction which spends the coin includes a valid $r$-round message for $P_j$

Rounds 3 … $\rho_{\pi} + 2$:
Execute $\text{GMW}(\pi)$ round-by-round so that in each round $r$ every party spends all its round $r$ referenced coins by a transaction which includes the round $r$ message in $\text{GMW}(\pi)$.
MPC with Robust Compens. : Construction

Idea: Use “GMW”-like compiler on the Ledger

GMW’:
Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Do nothing

Round 2:
Every $P_i$ commits to its input and broadcasts his view of the public setup.

Rounds 3 ... $\rho_\pi + 2$:
Execute $\pi$ round-by-round so that in each round every party proves (in NIZK) that the follows $\pi$

MPC with Robust Compensation
Round 0:
Setup generation (+ commitments to randomness)

Round 1:
Every party $P_i$ makes $n \cdot \rho_\pi$ special 1-coin transactions $B_{(i,j,r)}$:
- $P_j$ can spend coin in round $r$
- ref needs to have the protocol ID
- $R$ is true if the transaction which spends the coin includes a valid $r$-round message for $P_j$

Rounds 3 ... $\rho_\pi + 2$:
Execute $\text{GMW}(\pi)$ round-by-round so that in each round $r$ every party spends all its round $r$ referenced coins by a transaction which includes the round $r$ message in $\text{GMW}(\pi)$.
Security with Robust Compensation.

- **Case 2:** Some corrupted party does not make (consistent) transaction in Round 1
  - e.g. aborts or commits to a different setup.

  ... seems to have similar issue as before ...

- **Solution:** The validation predicate can be changed as:
  - Separates the parties into “islands” of consistent setups (depending on their Round-1 transactions).
  - For each island $I \subseteq [n]$: Compute the function among parties in $I$ (with all other parties’ input being 0)

- All honest parties are on the same island
- Corrupted parties can choose to play with the honest parties or participate in a computation independent of honest inputs.
Take Away Message and Open Directions
Take Away Message and Open Directions

Bitcoin opens new directions for cryptographic protocols

- Decentralized public ledger with inherent entropy
- Adding a reward/punishment mechanism restricts the set of likely attacks
- Limitations of crypto should be reconsidered (Impossibilities/Efficiency)
Take Away Message and Open Directions

Bitcoin opens new directions for cryptographic protocols

- Decentralized public ledger with inherent entropy
- Adding a reward/punishment mechanism restricts the set of likely attacks
- Limitations of crypto should be reconsidered (Impossibilities/Efficiency)

Future directions

- A game theoretic analysis might allow us to improve existing results … based on [GarayKatzMaurerTackmannZikas13]
- What more can we get from Blockchains?
Take Away Message and Open Directions

Bitcoin opens new directions for cryptographic protocols

- Decentralized public ledger with inherent entropy
- Adding a reward/punishment mechanism restricts the set of likely attacks
- Limitations of crypto should be reconsidered (Impossibilities/Efficiency)

Future directions

- A game theoretic analysis might allow us to improve existing results … based on [GarayKatzMaurerTackmannZikas13]
- What more can we get from Blockchains?

more in Jon’s talk tomorrow …