Bitcoin and Secure Computation With Money

How to Use Bitcoin to Play Internet Poker

Iddo Bentov
Technion

Ranjit Kumaresan
MIT

Tal Moran
IDC

GTACS @ BIU

January 8, 2015
Goals

MPC enhancements

- Impose fairness in MPC without an honest majority.
- Secure (reactive) MPC with money inputs and outputs
  - For example: poker.

Efficiency improvements to the MPC itself:

- Transform semi-honest secure MPC to MPC secure in the malicious setting, while penalizing caught deviations.
Formal model that incorporates coins

<table>
<thead>
<tr>
<th>Functionality $\mathcal{F}<em>\square$ versus functionality $\mathcal{F}</em>\star^*$ with coins</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>If party $P_i$ has (say) secret key $sk_i$ and sends it to party $P_j$, then both $P_i$ and $P_j$ will have the string $sk_i$.</strong></td>
</tr>
<tr>
<td><strong>If party $P_i$ has coins($x$) and sends $y &lt; x$ coins to party $P_j$, then $P_i$ will have coins($x - y$) and $P_j$ will have extra coins($y$).</strong></td>
</tr>
</tbody>
</table>
Formal model that incorporates coins

**Functionality \( F \) versus functionality \( F^* \) with coins**

- If party \( P_i \) has (say) secret key \( sk_i \) and sends it to party \( P_j \), then both \( P_i \) and \( P_j \) will have the string \( sk_i \).
- If party \( P_i \) has coins \((x)\) and sends \( y < x \) coins to party \( P_j \), then \( P_i \) will have coins \((x - y)\) and \( P_j \) will have extra coins \((y)\).

- Ideally, all the parties deem coins to be valuable assets.
- It is possible to define the *secure computation with coins* model directly, or with (UC) ideal functionalities.
- Sending coins \((x)\) may require a broadcast that reveals at least the amount \( x \) (not in zk-SNARK cryptocurrency like ZeroCash).
- We give proofs using the simulation paradigm (but not in this talk).
Claim-or-Refund for two parties $P_s, P_r$ (implicit in [Max11],[BBSU])

The $\mathcal{F}_{\text{CR}}^*$ Claim-or-Refund ideal functionality

1. The sender $P_s$ deposits (locks) her coins($q$) while specifying a timebound $\tau$ and a circuit $\phi(\cdot)$.
2. The receiver $P_r$ can claim (gain possession) of the coins($q$) by publicly revealing a witness $w$ that satisfies $\phi(w) = 1$.
3. If $P_r$ didn’t claim within time $\tau$, coins($q$) are refunded to $P_s$.

How to realize $\mathcal{F}_{\text{CR}}^*$ via Bitcoin

- The feature that is needed is “timelock” transactions.
- Technically: Bitcoin nodes agree to include a transaction with timelock field $\tau$ only if current block index/timestamp is $\geq \tau$
- It is possible to have more expressive schemes that allow not-yet-reached timelock transactions to reside on the blockchain (or local mempool), but this is prone to DoS.
$\mathcal{F}_{\text{CR}}^*$ via Bitcoin

<table>
<thead>
<tr>
<th>High-level description the $\mathcal{F}_{\text{CR}}^*$ implementation in Bitcoin</th>
</tr>
</thead>
<tbody>
<tr>
<td>- $P_s$ controls $TX_{\text{old}}$ that resides on the blockchain.</td>
</tr>
<tr>
<td>- $P_s$ creates a transaction $TX_{\text{new}}$ that spends $TX_{\text{old}}$ to a</td>
</tr>
<tr>
<td>Bitcoin script that can be redeemed by $P_s$ and $P_r$, or only by</td>
</tr>
<tr>
<td>$P_r$ by supplying a witness $w$ that satisfies $\phi(w) = 1$.</td>
</tr>
<tr>
<td>- $P_s$ asks $P_r$ to sign a timelock transaction that refunds $TX_{\text{new}}$</td>
</tr>
<tr>
<td>to $P_s$ at time $\tau$ (conditioned upon both $P_s$ and $P_r$ signing).</td>
</tr>
<tr>
<td>- After $P_r$ signs the refund, $P_s$ can safely broadcast $TX_{\text{new}}$.</td>
</tr>
</tbody>
</table>
Fiat-Star via Bitcoin

High-level description the $\mathcal{F}^*_\text{CR}$ implementation in Bitcoin

- $P_s$ controls $TX_{\text{old}}$ that resides on the blockchain.
- $P_s$ creates a transaction $TX_{\text{new}}$ that spends $TX_{\text{old}}$ to a Bitcoin script that can be redeemed by $P_s$ and $P_r$, or only by $P_r$ by supplying a witness $w$ that satisfies $\phi(w) = 1$.
- $P_s$ asks $P_r$ to sign a timelock transaction that refunds $TX_{\text{new}}$ to $P_s$ at time $\tau$ (conditioned upon both $P_s$ and $P_r$ signing).
- After $P_r$ signs the refund, $P_s$ can safely broadcast $TX_{\text{new}}$.

1. $P_s$ is safe because $P_r$ only sees $\text{Hash}(TX_{\text{new}})$, and therefore cannot broadcast $TX_{\text{new}}$ to cause $P_s$ to lose the coins.
2. $P_r$ can safely sign the random-looking data $\text{Hash}(TX_{\text{new}})$ because the protocol uses a freshly generated $(sk_R, pk_R)$ pair.
The structure of Bitcoin transactions

How standard Bitcoin transactions are chained

- $TX_{old} =$ earlier $TX$ output of coins $(q)$ is redeemable by $pk_A$
- $id_{old} =$ Hash($TX_{old}$)
- $PREPARE_{new} = (id_{old}, q, pk_B, 0)$  
  0 means no timelock
- $TX_{new} = (PREPARE_{new}, \text{Sign}_{sk_A}(PREPARE_{new}))$
- $id_{new} =$ Hash($TX_{new}$)
- Initial minting transaction specifies some $pk_M$ that belongs to a miner, and is created via proof of work.
Realization of $\mathcal{F}^*_CR$ via Bitcoin

The $\mathcal{F}^*_CR$ transaction

- $PREPARE_{\text{new}} = (id_{\text{old}}, q, (pk_S \land pk_R) \lor (\phi(\cdot) \land pk_R), 0)$
- $\phi(\cdot)$ can be $\text{SHA256}(\cdot) == Y$ where $Y$ is hardcoded.
- $TX_{\text{new}} = (PREPARE_{\text{new}}, \text{Sign}_{sk_S}(PREPARE_{\text{new}}))$
- $id_{\text{new}} = \text{Hash}(TX_{\text{new}})$
- $P_s$ sends $PREPARE_{\text{refund}} = (id_{\text{new}}, q, pk_S, \tau)$ to $P_r$
- $P_r$ sends $\sigma_R = \text{Sign}_{sk_R}(PREPARE_{\text{refund}})$ to $P_s$
- $P_s$ broadcasts $TX_{\text{new}}$ to the Bitcoin network
- If $P_r$ doesn’t reveal $w$ until time $\tau$ then $P_s$ creates $TX_{\text{refund}} = (PREPARE_{\text{refund}}, (\text{Sign}_{sk_S}(PREPARE_{\text{refund}}), \sigma_R))$ and broadcasts it to reclaim her $q$ coins
Definition of fair secure multiparty computation with penalties

- An honest party never has to pay any penalty
- If a party aborts after learning the output and doesn’t deliver output to honest parties $\Rightarrow$ every honest party is compensated
**Definition of fair secure multiparty computation with penalties**

- An honest party never has to pay any penalty
- If a party aborts after learning the output and doesn’t deliver output to honest parties $\Rightarrow$ every honest party is compensated

**Outline of $F^*_f$ – fairness with penalties for any function $f$**

- $P_1, \ldots, P_n$ run secure unfair MPC for $f(x_1, \ldots, x_n)$ that
  1. Computes shares $(y_1, \ldots, y_n)$ of the output $y = f(x_1, \ldots, x_n)$
  2. Computes $Tags = (\text{com}(y_1), \ldots, \text{com}(y_n)) = (\text{hash}(y_1), \ldots, \text{hash}(y_n))$
  3. Delivers $(y_i, Tags)$ to every $P_i$

- $P_1, \ldots, P_n$ deposit coins and run fair reconstruction (fair exchange) with penalties to swap the $y_i$’s among themselves.
Fair exchange in the $\mathcal{F}_{CR}^*$-hybrid model - the ladder construction

“Abort” attack:

$P_2$ claims without depositing

$P_1 \xrightarrow{w_2, q, \tau} P_2$

$P_2 \xrightarrow{w_1, q, \tau} P_1$

Malicious coalition:

Coalition $P_1, P_2$ obtains $w_3$ from $P_3$

$P_2$ doesn’t claim the top transaction $P_3$ isn’t compensated

$P_1 \xrightarrow{w_1 \land w_2, q, \tau_2} P_2$

$P_2 \xrightarrow{w_1, q, \tau_1} P_1$

$\tau_1 < \tau_2$

$P_1 \xrightarrow{w_1 \land w_2 \land w_3, q, \tau_3} P_2$

$P_2 \xrightarrow{w_1 \land w_3, q, \tau_2} P_3$

$P_3 \xrightarrow{w_1, q, \tau_1} P_1$

$\tau_1 < \tau_2 < \tau_3$
Fair exchange in the $\mathcal{F}^*_{CR}$-hybrid model - the ladder construction

“Abort” attack:
$P_2$ claims without depositing

Fair exchange:
$P_1$ claims by revealing $w_1$
$\Rightarrow P_2$ can claim by revealing $w_2$
**Fair exchange in the $\mathcal{F}^*_\text{CR}$-hybrid model - the ladder construction**

**“Abort” attack:**

$P_2$ claims without depositing

**Fair exchange:**

$P_1$ claims by revealing $w_1$

$\Rightarrow P_2$ can claim by revealing $w_2$

**Malicious coalition:**

Coalition $P_1, P_2$ obtain $w_3$ from $P_3$

$P_2$ doesn’t claim the top transaction

$P_3$ isn’t compensated
Fair exchange in the $\mathcal{F}_{CR}^*$-hybrid model - the ladder construction (contd.)

Fair exchange:

- Bottom two levels: $P_1, P_2$ get compensated by $P_3$
- Top two levels: $P_3$ gets her refunds by revealing $w_3$

Full ladder:
In principle, jointly locking coins for fair exchange can work well:

1. \( M = \text{"if } P_1, P_2, P_3, P_4 \text{ sign this message with inputs of coins}(3x) \text{ each then their } 3x \text{ coins are locked into 4 outputs of coins}(3x) \text{ each, where each } P_i \text{ can redeem output } T_i \text{ with a witness } w_i \text{ that satisfies } \phi_i, \text{ and after time } \tau \text{ anyone can divide an unredeemed output } T_i \text{ equally to } \{P_1, P_2, P_3, P_4\} \setminus \{P_i\}"\)

2. \( P_1, P_2, P_3, P_4 \text{ sign } M \text{ and broadcast it, and after } M \text{ is confirmed, each } P_i \text{ redeems coins}(x) \text{ by revealing } w_i \)
Due to a design flaw, to implement $\mathcal{F}_{\text{ML}}^\star$ in the current Bitcoin protocol an \textit{unfair} secure MPC needs to be invoked, where the input of $P_i$ is $inp_i = \text{Sign}_{sk_i}(\text{PREPARE}_{\text{lock}})$, and the output to all parties is $\text{SHA256d}(\text{PREPARE}_{\text{lock}}, inp_1, \ldots, inp_n)$. 
Practicality of multiparty fair exchange with penalties in Bitcoin

- Due to a design flaw, to implement $\mathcal{F}_{\text{ML}}^*$ in the current Bitcoin protocol an *unfair* secure MPC needs to be invoked, where the input of $P_i$ is $\text{inp}_i = \text{Sign}_{sk_i}(\text{PREPARE}_{\text{lock}})$, and the output to all parties is $\text{SHA256d}(\text{PREPARE}_{\text{lock}}, \text{inp}_1, \ldots, \text{inp}_n)$.

- $\mathcal{F}_{\text{ML}}^*$ requires $O(1)$ Bitcoin rounds and $O(n^2)$ transaction data (and $O(n^2)$ signature operations), while the ladder requires $O(n)$ Bitcoin rounds and $O(n)$ transactions data.
Practicality of multiparty fair exchange with penalties in Bitcoin

- Due to a design flaw, to implement $\mathcal{F}_{\text{ML}}^*$ in the current Bitcoin protocol an *unfair* secure MPC needs to be invoked, where the input of $P_i$ is $inp_i = \text{Sign}_{sk_i}(\text{PREPARE}_{\text{lock}})$, and the output to all parties is $\text{SHA256d}(\text{PREPARE}_{\text{lock}}, inp_1, \ldots, inp_n)$.

- $\mathcal{F}_{\text{ML}}^*$ requires $O(1)$ Bitcoin rounds and $O(n^2)$ transaction data (and $O(n^2)$ signature operations), while the ladder requires $O(n)$ Bitcoin rounds and $O(n)$ transactions data.

Recap:

- Multiparty fair computation can be implemented in Bitcoin via the ladder construction.

- Multiparty fair computation can be implemented in Bitcoin via $\mathcal{F}_{\text{ML}}^*$ with one superfluous unfair MPC.

- Multiparty fair computation can be implemented via $\mathcal{F}_{\text{ML}}^*$ directly with an enhanced Bitcoin protocol.
Comparison with other ways to achieve fairness

Gradual release

- Even with only 2 parties, the number of rounds depends on a security parameter.

Trusted bank

Legally Enforceable Fairness in Secure Two-Party Computation [Lindell 2008]

Requires a trusted party to provide an ideal bank functionality.

Bank balance of a party can go negative? Bounced cheques?

2-party only: the bank can provide $F^{\star CR}$ or $F^{\star ML}$ to use our constructions directly, or implement similar protocols.

Disadvantage of Bitcoin: funny money?
Comparison with other ways to achieve fairness

Gradual release

- Even with only 2 parties, the number of rounds depends on a security parameter.
- With Bitcoin, the PoW miners do all the heavy lifting.
Comparison with other ways to achieve fairness

**Gradual release**
- Even with only 2 parties, the number of rounds depends on a security parameter.
- With Bitcoin, the PoW miners do all the heavy lifting.

**Trusted bank**
- Legally Enforceable Fairness in Secure Two-Party Computation [Lindell 2008]
Comparison with other ways to achieve fairness

**Gradual release**
- Even with only 2 parties, the number of rounds depends on a security parameter.
- With Bitcoin, the PoW miners do all the heavy lifting.

**Trusted bank**
- Legally Enforceable Fairness in Secure Two-Party Computation [Lindell 2008]
- Requires a trusted party to provide an ideal bank functionality.
Comparison with other ways to achieve fairness

Gradual release
- Even with only 2 parties, the number of rounds depends on a security parameter.
- With Bitcoin, the PoW miners do all the heavy lifting.

Trusted bank
- Legally Enforceable Fairness in Secure Two-Party Computation [Lindell 2008]
- Requires a trusted party to provide an ideal bank functionality.
- Bank balance of a party can go negative? Bounced cheques?
Comparison with other ways to achieve fairness

Gradual release

- Even with only 2 parties, the number of rounds depends on a security parameter.
- With Bitcoin, the PoW miners do all the heavy lifting.

Trusted bank

- Legally Enforceable Fairness in Secure Two-Party Computation [Lindell 2008]
- Requires a trusted party to provide an ideal bank functionality.
- Bank balance of a party can go negative? Bounced cheques?
- 2-party only: the bank can provide $\mathcal{F}^*_\text{CR}$ or $\mathcal{F}^*_\text{ML}$ to use our constructions directly, or implement similar protocols.
Comparison with other ways to achieve fairness

**Gradual release**
- Even with only 2 parties, the number of rounds depends on a security parameter.
- With Bitcoin, the PoW miners do all the heavy lifting.

**Trusted bank**
- Legally Enforceable Fairness in Secure Two-Party Computation [Lindell 2008]
- Requires a trusted party to provide an ideal bank functionality.
- Bank balance of a party can go negative? Bounced cheques?
- 2-party only: the bank can provide $\mathcal{F}_{\text{CR}}^*$ or $\mathcal{F}_{\text{ML}}^*$ to use our constructions directly, or implement similar protocols.
- Disadvantage of Bitcoin: funny money?
Secure cash distribution and poker

How to Use Bitcoin to Play Internet Poker

Iddo Bentov
Technion

Ranjit Kumaresan
MIT

Tal Moran
IDC
“Paradoxical” Abilities 1983-

- Exchanging Secret Messages without Ever Meeting
- Simultaneous Contract Signing Over the Phone
- Generating exponentially long pseudo random strings indistinguishable from random
- Proving a theorem without revealing the proof
- Playing any digital game without referees
- Private Information Retrieval
Secure cash distribution with penalties

Ideal 2-party secure (non-reactive) cash distribution functionality:

1. Wait to receive \((x_1, \text{coins}(d_1))\) from \(P_1\) and \((x_2, \text{coins}(d_2))\) from \(P_2\).
2. Compute \((y, v) \leftarrow f(x_1, x_2, d_1, d_2)\).
3. Send \((y, \text{coins}(v))\) to \(P_1\) and \((y, \text{coins}(d_1 + d_2 - v))\) to \(P_2\).
Secure cash distribution with penalties

Ideal 2-party secure (non-reactive) cash distribution functionality:

1. Wait to receive \((x_1, \text{coins}(d_1))\) from \(P_1\) and \((x_2, \text{coins}(d_2))\) from \(P_2\).
2. Compute \((y, v) \leftarrow f(x_1, x_2, d_1, d_2)\).
3. Send \((y, \text{coins}(v))\) to \(P_1\) and \((y, \text{coins}(d_1 + d_2 - v))\) to \(P_2\).

- In the general case, each party \(P_i\) has input \((x_i, \text{coins}(d_i))\) and receives output \((y, \text{coins}(v_i))\).
- Use-cases: generalized lottery, incentivized computation, ...
Blackbox secure cash distribution

- Blackbox realization of secure cash distribution in the $\mathcal{F}^*_{\text{CR}}$-hybrid model.
- Assume that input coins amount of $P_i$ is $m_i$-bit number.

Step 1: commit to random secrets (preprocessing)

Invoke secure MPC where all $i \in [n], j \in [n] \setminus \{i\}, k \in [m_i]$:

- $P_i$ picks random witness $w_{i,j,k} \leftarrow \{0, 1\}^\lambda$ (also random $r_{i,j,k}$).
- $P_i$ computes $c_{i,j,k} \leftarrow \text{commit}(1^\lambda, w_{i,j,k}, r_{i,j,k})$.
- $P_i$ $n$-out-of-$n$ secret shares each witness $w_{i,j,k}$.
- $P_i$ outputs $c_{i,j,k}$ and the $i$-th share of each $w_{i,j,k}$ to each $P_j$.

Then, each $P_i$ makes $\mathcal{F}^*_{\text{CR}}$ transaction $P_i \xrightarrow{w_{i,j,k}} 2^k, \tau P_j$.
Blackbox secure cash distribution (contd.)

Assume that the input coin amounts is $d = (d_1, \ldots, d_n)$ and the string inputs are $(x_1, x_2, \ldots, x_n)$.

Step 2: compute the cash distribution

Invoke secure MPC (unfair for now) for the cash distribution:

- Compute the output coin amounts $v = (v_1, v_2, \ldots, v_n)$.
- Derive numbers $b_{i,j}$ that specify how many coins $P_i$ needs to send $P_j$ according to the input coins $d$ and output coins $v$.
- Let $(b_{i,j,1}, b_{i,j,2}, \ldots, b_{i,j,m_i})$ be the binary expansion of $b_{i,j}$.
- For all $i, j, k$, if $b_{i,j,k} = 1$ then reconstruct $w_{i,j,k}$ and concatenate it to the output.
- Compute $y = f(x_1, x_2, \ldots, x_n)$ and output $y$ too.

Then, use fair exchange with penalties (with time limit $< \tau$) to deliver the output to all parties, so that $\mathcal{F}_{\text{CR}}^*$ claims will ensue.
Reactive secure cash distribution

Ingredients needed:

- See-saw instead of the ladder construction, to force parties to make the next move.
Reactive secure cash distribution

Ingredients needed:

- See-saw instead of the ladder construction, to force parties to make the next move.
- The given secure MPC (whitebox) where for every round $j$ a single message is broadcast by a designated party $P_{ij}$.
Reactive secure cash distribution

Ingredients needed:

- See-saw instead of the ladder construction, to force parties to make the next move.
- The given secure MPC (whitebox) where for every round \( j \) a single message is broadcast by a designated party \( P_{ij} \).
- \( \mathcal{F}_{\text{CR}}^* \) transactions \( P_i \xrightarrow{\phi_{i,j}} q,\tau P_j \) where \( \phi_{i,j} \) is a circuit (script) that is satisfied if \( P_i \) create multiple signed extensions of protocol’s execution (with a unique starting nonce).
Reactive secure cash distribution

Ingredients needed:

- See-saw instead of the ladder construction, to force parties to make the next move.
- The given secure MPC (whitebox) where for every round $j$ a single message is broadcast by a designated party $P_{ij}$.
- $F^*_CR$ transactions $P_i \xrightarrow{\phi_{i,j}} q,\tau P_j$ where $\phi_{i,j}$ is a circuit (script) that is satisfied if $P_i$ create multiple signed extensions of protocol’s execution (with a unique starting nonce).
- Blackbox secure cash distribution as described, with refunds at time $\tau$ that exceeds the see-saw time limits, and hence with circuits specified at start that are checked in the final rounds.
The see-saw construction: 2 parties

**Roof deposit.**

\[ P_1 \xrightarrow{TT_{m,2}} P_2 \]

\[ P_1 \xrightarrow{q,\tau_{m,2}} P_2 \]

\( TX_{m,2} \)

**See-saw deposits.** For \( r = m - 1 \) to 1:

\[ P_2 \xrightarrow{TT_{r+1,1}} P_1 \]

\[ P_2 \xrightarrow{2q,\tau_{r+1,1}} P_1 \]

\( TX_{r+1,1} \)

\[ P_1 \xrightarrow{TT_{r,2}} P_2 \]

\[ P_1 \xrightarrow{2q,\tau_{r,2}} P_2 \]

\( TX_{r,2} \)

**Floor deposit.**

\[ P_2 \xrightarrow{TT_{1,1}} P_1 \]

\[ P_2 \xrightarrow{q,\tau_{1,1}} P_1 \]

\( TX_{1,1} \)
The see-saw construction: multiparty

ROOF DEPOSITS. For each $j \in [n - 1]$:

$$P_j \xrightarrow{\text{TT}_n} P_n$$

LADDER DEPOSITS. For $i = n - 1$ down to 2:

- **Rung unlock:** For $j = n$ down to $i + 1$:
  $$P_j \xrightarrow{\text{TT}_i \land U_{i,j}} P_i$$

- **Rung climb:**
  $$P_{i+1} \xrightarrow{\text{TT}_i} P_i$$

- **Rung lock:** For each $j = n$ down to $i + 1$:
  $$P_i \xrightarrow{\text{TT}_{i-1} \land U_{i,j}} P_j$$

FOOT DEPOSIT.

$$P_2 \xrightarrow{\text{TT}_1} P_1$$
The see-saw construction: multiparty (contd.)

Properties of the multiparty see-saw

- Quadratic round complexity (ladder is linear).
The see-saw construction: multiparty (contd.)

Properties of the multiparty see-saw

- Quadratic round complexity (ladder is linear).
- Party $P_i$ who aborts pays compensation to all other parties.
Properties of the multiparty see-saw

- Quadratic round complexity (ladder is linear).
- Party $P_i$ who aborts pays compensation to all other parties.
- In the ladder $P_i$ can abort and then nobody learns the secret.
Properties of the multiparty see-saw

- Quadratic round complexity (ladder is linear).
- Party $P_i$ who aborts pays compensation to all other parties.
- In the ladder $P_i$ can abort and then nobody learns the secret.
- This is crucial for reactive functionalities:
  - Consider poker: suppose that in round $j$ all parties exchange shares to reveal the top card on the deck.
  - If $P_i$ didn’t like this top card, we must not allow $P_i$ to abort in round $j + 1$ without punishment.
## Properties of the multiparty see-saw

- Quadratic round complexity (ladder is linear).
- Party $P_i$ who aborts pays compensation to all other parties.
- In the ladder $P_i$ can abort and then nobody learns the secret.
- This is crucial for reactive functionalities:
  - Consider poker: suppose that in round $j$ all parties exchange shares to reveal the top card on the deck.
  - If $P_i$ didn’t like this top card, we must not allow $P_i$ to abort in round $j + 1$ without punishment.
- The circuits verify a signed extension of the entire execution transcript, and that this extension conforms with the protocol.
The see-saw construction: multiparty (contd.)

Properties of the multiparty see-saw

- Quadratic round complexity (ladder is linear).
- Party $P_i$ who aborts pays compensation to all other parties.
- In the ladder $P_i$ can abort and then nobody learns the secret.
- This is crucial for reactive functionalities:
  - Consider poker: suppose that in round $j$ all parties exchange shares to reveal the top card on the deck.
  - If $P_i$ didn’t like this top card, we must not allow $P_i$ to abort in round $j + 1$ without punishment.
- The circuits verify a signed extension of the entire execution transcript, and that this extension conforms with the protocol.
- $\Rightarrow$ needs more expressive scripting language than vanilla Bitcoin, but not Turing complete scripts because the round bounds are known in advance.
The see-saw construction: poker

- No need to run reactive secure MPC that corresponds to rounds of the see-saw.
The see-saw construction: poker

- No need to run reactive secure MPC that corresponds to rounds of the see-saw.

- Invoke (preprocess) at start an unfair SFE that:
  - Shuffles the deck according to the parties’ random inputs.
  - Computes commitments to shares of all the cards.
  - Deals shares of the hands and shares of the rest of the cards to all parties, and also delivers all the commitments to all parties.
The see-saw construction: poker

- No need to run reactive secure MPC that corresponds to rounds of the see-saw.
- Invoke (preprocess) at start an unfair SFE that:
  - Shuffles the deck according to the parties’ random inputs.
  - Computes commitments to shares of all the cards.
  - Deals shares of the hands and shares of the rest of the cards to all parties, and also delivers all the commitments to all parties.
- Make the cash distribution transactions whose circuits verify the signatures of a transcript, then scan it while performing arithmetic calculations.
The see-saw construction: poker

- No need to run reactive secure MPC that corresponds to rounds of the see-saw.
- Invoke (preprocess) at start an unfair SFE that:
  - Shuffles the deck according to the parties’ random inputs.
  - Computes commitments to shares of all the cards.
  - Deals shares of the hands and shares of the rest of the cards to all parties, and also delivers all the commitments to all parties.
- Make the cash distribution transactions whose circuits verify the signatures of a transcript, then scan it while performing arithmetic calculations.
- The $\mathcal{F}_{\text{CR}}^{\star}$ circuit in each round of the see-saw will verify signatures of a transcript, then enforce betting rules or expect a party to reveal a share of a card.
- For example: if all partied called and the top card on the deck should be revealed, then the next see-saw circuits will require each party to reveal her share of the top card.
Some open questions

- Lower bound of linear number of rounds for fairness with penalties in the $\mathcal{F}_{CR}^*$-hybrid model?
- Bounds for the minimal deposit amounts? Rational analysis?
- Constructing secure cash distribution with penalties from blackbox secure MPC and $\mathcal{F}_{CR}^*$?
Some open questions

- Lower bound of linear number of rounds for fairness with penalties in the $F^*_\text{CR}$-hybrid model?
- Bounds for the minimal deposit amounts? Rational analysis?
- Constructing secure cash distribution with penalties from blackbox secure MPC and $F^*_\text{CR}$?

Thank you.