How Fair is Your Protocol?
A Utility-based Approach to Protocol Optimality

[PODC 2015]

Juan Garay (Yahoo Labs)
Jonathan Katz (UMD)
Björn Tackmann (UCSD)
Vassilis Zikas (ETH Zürich)
Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B
# Two Coin-Toss Protocols

<table>
<thead>
<tr>
<th>Protocol A</th>
<th>Protocol B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Each party commits to a bit.</td>
<td>1. Each party commits to a bit.</td>
</tr>
<tr>
<td>2. Both parties open their commitments.</td>
<td></td>
</tr>
<tr>
<td>3. The result is the XOR.</td>
<td></td>
</tr>
</tbody>
</table>
Two Coin-Toss Protocols

Protocol A

1. Each party commits to a bit.
2. Both parties open their commitments.
3. The result is the XOR.

Protocol B

1. Each party commits to a bit.
2. They toss a coin $i^* \in \{1,2\}$.
Two Coin-Toss Protocols

<table>
<thead>
<tr>
<th>Protocol A</th>
<th>Protocol B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Each party commits to a bit.</td>
<td>1. Each party commits to a bit.</td>
</tr>
<tr>
<td>2. Both parties open their commitments.</td>
<td>2. They toss a coin $i^* \in {1,2}$.</td>
</tr>
<tr>
<td>3. The result is the XOR.</td>
<td>3. $p(3-i^<em>)$ opens its commitment to $p_{i^</em>}$</td>
</tr>
</tbody>
</table>
# Two Coin-Toss Protocols

<table>
<thead>
<tr>
<th>Protocol A</th>
<th>Protocol B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Each party commits to a bit.</td>
<td>1. Each party commits to a bit.</td>
</tr>
<tr>
<td>2. Both parties open their commitments.</td>
<td>2. They toss a coin $i^* \in {1,2}$.</td>
</tr>
<tr>
<td>3. The result is the XOR.</td>
<td>3. $p_{3-i^<em>}$ opens its commitment to $p_{i^</em>}$.</td>
</tr>
<tr>
<td></td>
<td>4. $p_{i^<em>}$ opens its commitment to $p_{(3-i^</em>)}$.</td>
</tr>
</tbody>
</table>
## Two Coin-Toss Protocols

<table>
<thead>
<tr>
<th>Protocol A</th>
<th>Protocol B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Each party commits to a bit.</td>
<td>1. Each party commits to a bit.</td>
</tr>
<tr>
<td>2. Both parties open their commitments.</td>
<td>2. They toss a coin $i^* \in {1,2}$.</td>
</tr>
<tr>
<td>3. The result is the XOR.</td>
<td>3. $p_{(3-i^<em>)}$ opens its commitment to $p_{i^</em>}$.</td>
</tr>
<tr>
<td></td>
<td>4. $p_{i^<em>}$ opens its commitment to $p_{(3-i^</em>)}$.</td>
</tr>
<tr>
<td></td>
<td>5. The result is the XOR.</td>
</tr>
</tbody>
</table>
Fairness in SFE

Fairness:
- “if one party learns the output, the other party also learns it,”
- generally impossible in 2PC [Cleve, STOC’86].
Fairness in SFE

\[ z = f(x, y) \]

- does not get z
- does not get z
- does not get z
- gets z
- gets z
- does not get z
- gets z
Fairness in SFE

Possible outcomes (intuitively):

- $x$ does not get $z$
- $y$ does not get $z$
- $x$ does not get $z$
- $y$ gets $z$
- $x$ gets $z$
- $y$ does not get $z$
- $x$ gets $z$
- $y$ gets $z$
Possible outcomes (intuitively):

<table>
<thead>
<tr>
<th></th>
<th>“Fairness”</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not get z</td>
<td>good</td>
</tr>
<tr>
<td>gets z</td>
<td>bad</td>
</tr>
<tr>
<td>does not get z</td>
<td>good</td>
</tr>
<tr>
<td>gets z</td>
<td>good</td>
</tr>
</tbody>
</table>

Fairness in SFE

\[ z = f(x, y) \]
Fairness in SFE

Possible outcomes (intuitively):

<table>
<thead>
<tr>
<th></th>
<th>“Fairness”</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not get z</td>
<td>does not get z</td>
<td>good</td>
</tr>
<tr>
<td>does not get z</td>
<td>gets z</td>
<td>bad</td>
</tr>
<tr>
<td>gets z</td>
<td>does not get z</td>
<td>good</td>
</tr>
<tr>
<td>gets z</td>
<td>gets z</td>
<td>good</td>
</tr>
</tbody>
</table>
Possible outcomes (intuitively):

<table>
<thead>
<tr>
<th></th>
<th>&quot;Fairness&quot;</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not get $z$</td>
<td>does not get $z$</td>
<td>good</td>
</tr>
<tr>
<td>$\gamma_{00}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>does not get $z$</td>
<td>gets $z$</td>
<td>bad</td>
</tr>
<tr>
<td>$\gamma_{10}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gets $z$</td>
<td>does not get $z$</td>
<td>good</td>
</tr>
<tr>
<td>$\gamma_{01}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gets $z$</td>
<td>gets $z$</td>
<td>good</td>
</tr>
<tr>
<td>$\gamma_{11}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Natural conditions: $\gamma_{01} < \gamma_{00}, \gamma_{11}$ and $\gamma_{00}, \gamma_{11} < \gamma_{10}$
Fairness in SFE

\[ z = f(x, y) \]

Protocol comparison and optimality:

- the utilities for the individual outcomes define an expected payoff for each adversarial strategy,
- a protocol is better (fairer) if the expected payoff of the best adversarial strategy is smaller.

Natural conditions: \( \gamma_{01} < \gamma_{00}, \gamma_{11} \) and \( \gamma_{00}, \gamma_{11} < \gamma_{10} \)

Rational fairness [Asharov-Canetti-Hazay, 2011]

1/p-Security [Gordon-Katz, 2010; …]
Rational Protocol Design

- Two-move “meta” game,
- zero-sum: $u_D = -u_A$,
- $\epsilon$-subgame-perfect equilibrium.

[Garay-Katz-Maurer-T-Zikas, FOCS 2013]
Rational Protocol Design

\[ F \]
Step 1: Relax functionality
Step 1: Relax functionality
Rational Protocol Design

Step 1: Relax functionality

\[ \exists S: \]
Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*
Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in ideal

s.t. ∃ S:

Event: Ideal adversary exploits fault.
Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

Step 3: Define payoff
Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in *ideal*

Event: Ideal adversary exploits fault.

Step 3: Define payoff

Assign a payoff to each event.
Rational Protocol Design

Step 1: Relax functionality

Step 2: Define events in ideal

Event: Ideal adversary exploits fault.

Assign a payoff to each event.

Step 3: Define payoff

Payoff(\mathcal{A}) = \min_{S \text{ "good"}} \text{payoff}(S)
Defining Fairness (1)

1. Get inputs $x$ and $y$
2. Compute $z = f(x,y)$
3. Possibly: Output $z$ to $p_1$ and $p_2$

$F_{\text{fair-sfe}}$
Step 1:

1. Get inputs $x$ and $y$
2. Compute $z = f(x, y)$
3. Output $z$ to $p_1$ and $p_2$
Defining Fairness (3)

The protocol $\pi$ realizes $F_{\text{unfair-sfe}}$, i.e., there is $S$:

1. Get inputs $x$ and $y$
2. Compute $z = f(x, y)$
3. Output $z$ to $p_1$ and $p_2$
Defining Fairness (4)

Step 2: Define events in the ideal execution:

(a) Neither party gets the output: $E_{00}$, payoff $\gamma_{00}$
(b) Only honest party gets the output: $E_{01}$, payoff $\gamma_{01}$
(c) Only corrupted party gets the output: $E_{10}$, payoff $\gamma_{10}$
(d) Both parties get the output: $E_{11}$, payoff $\gamma_{11}$

Natural conditions: $\gamma_{01} < \gamma_{00}$, $\gamma_{11}$ and $\gamma_{00}$, $\gamma_{11} < \gamma_{10}$
Defining Fairness (5)

Step 3: Define the expected payoff for each $S$:

$$\text{payoff}(S) = \sum_{i,j \in \{0,1\}} \Pr(E_{ij}) \cdot \gamma_{ij}$$

The payoff of an adversary is the expected payoff of the best simulator:

$$\text{Payoff}(\mathcal{A}) = \min_{\text{"good" } S} \text{payoff}(S)$$
Optimal Protocol for Two Party SFE

- The protocol achieves $\frac{\gamma_{10} + \gamma_{11}}{2}$.
- This is optimal (see next slide).

1. In an unfair SFE:
   (a) choose $i^* \in \{1,2\}$
   (b) compute a sharing of the output value
   (c) output $i^*$ and one share to each party
2. in case of abort, restart with default input for other party
3. $p_{(3-i^*)}$ sends its share to $p_{i^*}$
4. $p_{i^*}$ sends its share to $p_{(3-i^*)}$
Optimal Protocol for Two Party SFE

- The protocol achieves $\frac{\gamma_{10} + \gamma_{11}}{2}$.
- This is optimal (see next slide).

Proof idea:

- secure w/o fairness (based on underlying SFE and repeat before leaking output)
- the simulator chooses $i^*$ uniformly at random

1. In an unfair SFE:
   (a) choose $i^* \in \{1,2\}$
   (b) compute a sharing of the output value
   (c) output $i^*$ and one share to each party
2. in case of abort, restart with default input for other party
3. $p_{(3-i^*)}$ sends its share to $p_{i^*}$
4. $p_{i^*}$ sends its share to $p_{(3-i^*)}$
Optimal Protocol for Two Party SFE

There exist functions such that...

Round 1
Round 2
...
Round n
Optimal Protocol for Two Party SFE

There exist functions such that...

Run the honest protocol as follows.

In each round:
• receive the honest party’s message,
• check whether the honest protocol would generate output,
• if so, then abort,
• otherwise, send the honestly computed message for this round
Optimal Protocol for Two Party SFE

Run the honest protocol as follows.

In each round:
• receive the honest party’s message,
• check whether the honest protocol would generate output,
• if so, then abort,
• otherwise, send the honestly computed message for this round.

There exist functions such that...

In each round:
• $p_1$ receives the output first, or
• $p_2$ receives the output first, or
• both receive the output.
Optimal Protocol for Two Party SFE

Run the honest protocol as follows.

In each round:
• receive the honest party’s message,
• check whether the honest protocol would generate output,
• if so, then abort,
• otherwise, send the honestly computed message for this round.

In each round:
• \( p_1 \) receives the output first, or
• \( p_2 \) receives the output first, or
• both receive the output.

Proof idea:
• a protocol can be improved by never outputting to both in the same round
• at least one party is first with probability at least \( 1/2 \)

There exist functions such that...
The Multi-Party Case
The Multi-Party Case
The Multi-Party Case
The Multi-Party Case

$F_{\text{fair-sfe}}$

1. Get inputs $x_1, x_2, ..., x_n$
2. Compute $z = f(x_1, x_2, ..., x_n)$
3. Possibly: Output $z$ to $p_1, p_2, ..., p_n$
The Multi-Party Case

$F_{\text{fair-sfe}}$

1. Get inputs $x_1, x_2, \ldots, x_n$
2. Compute $z = f(x_1, x_2, \ldots, x_n)$
3. Possibly: Output $z$ to $p_1, p_2, \ldots, p_n$

Give output only to me!
Multi-Party Fairness

Step 2: Define events in the *ideal* execution:

A. No party gets the output: $E_{00}$, payoff $\gamma_{00}$

B. Exactly all honest parties get the output: $E_{01}$, payoff $\gamma_{01}$

C. Not all honest parties, but some corrupted party gets the output: $E_{10}$, payoff $\gamma_{10}$

D. All honest parties and some corrupted party get the output: $E_{11}$, payoff $\gamma_{11}$

Here: stronger condition $\gamma_{01} < \gamma_{00} < \gamma_{11} < \gamma_{10}$. 
Multi-Party Fairness

payoff

n/2

# corrupted parties
Multi-Party Fairness

Standard SFE protocols

payoff

n/2

# corrupted parties
Multi-Party Fairness

Standard SFE protocols

“release to random party”

payoff

\( n/2 \)

\# corrupted parties
Multi-Party Fairness

Standard SFE protocols

Rough idea: Give the output to some party, let him distribute.
Other Relaxed Notions of Fairness


❖ Rational fairness [Asharov-Canetti-Hazay, 2011]
  ❖ Not closely related, after all…

  ❖ Similar (quantitative) guarantee,
  ❖ protocols for functions with small domain or range,
  ❖ formally more relaxed definition.
Rational Protocol Design

- General framework (beyond fairness),
- supports composition (via the underlying framework),
- generalizes to reactive functionalities (follow-up).

cf. Ranjit’s talk
Rational Protocol Design

“Rational” commitment*:

1. Get input $x$
2. Output $x$

$F_{\text{commit}}$

* as mentioned by Rosario on Monday.
Rational Protocol Design

“Rational” commitment*: 

\[ F_{\text{commit}} \]

1. Get input \( x \)
2. Output \( x \)

“open”

“received value”

* as mentioned by Rosario on Monday.
Rational Protocol Design

“Rational” commitment*:

1. Get input $x$
2. Output $x$

* as mentioned by Rosario on Monday.
Summary

- RPD is a general framework capturing incentives,
- idea: build the best protocol w.r.t. the incentives,
- we showed optimal protocols for fairness in SFE.
- Follow-up: Reactive functionalities.