How to Solve Any Protocol Problem
(or How to Play Any Mental Game)

by

Oded Goldreich

Computer Science Dept.
Technion, Israel

Based on works with:

(1) Micali and Wigderson
(2) Vainish
MOTIVATION

\[ f: \mathbb{D} \times \mathbb{D} \times \cdots \times \mathbb{D} \]

\( n \)-ARY FUNCTION

\[
\begin{align*}
\frac{P_1}{x_1} & \quad \frac{P_2}{x_2} & \quad \ldots & \quad \frac{P_n}{x_n} \\
\hline
f(x_1, \ldots, x_n) & \quad f(x_1, \ldots, x_n) & \quad f(x_1, \ldots, x_n)
\end{align*}
\]
Motivation

\[ f: \mathbb{D} \times \mathbb{D} \times \ldots \times \mathbb{D} \]

n-ary function

\[ P_i \]
\[ x_i \]

\[ P_2 \]
\[ x_2 \]

\[ \ldots \]

\[ P_n \]
\[ x_n \]

\[ f(x_1, \ldots, x_n) \]

\[ f(x_1, \ldots, x_n) \]

\[ f(x_1, \ldots, x_n) \]

TRUSTED 3RD PARTY
Motivation

\[ f : \mathbb{D} \times \mathbb{D} \times \ldots \times \mathbb{D} \]

n-ary function

\[ P_1 \quad P_2 \quad \ldots \quad P_n \]

\[ x_1 \quad x_2 \quad x_n \]

TRUSTED 3RD PARTY

\[ f(x_1 \ldots x_n) \quad f(x_1 \ldots x_n) \quad f(x_1 \ldots x_n) \]

Issues:

1. Correctness
2. Privacy

\{ SIM. COMMIT. FORCING PROPER EXEC. \}
FORMAL SETTING - PRELIMINARIES

- A protocol problem is an \( n \)-ary function, \( f \).

- Model (for solutions)

  A computation is called efficient if it is completed in time polynomial in the complexity of \( f \).

- A solution to the protocol problem \( f \) is an efficient protocol guaranteeing "simultaneous commitment", "correctness", and "privacy" in presence of \( \leq \frac{\sqrt{n}}{2} \) faulty but feasible processors.
A solution to the protocol problem is an efficient fault-tolerant protocol guaranteeing (w.r.t $\frac{n}{2}$ feasible faults):

1. Simultaneous commitment to initial values.

2. (Correspondingly) correct output values.

3. Maximum privacy of the initial values.

Example:

$$\sum_{i=1}^{n} x_i$$

[Coh.]
Our Result

Assuming existence of public-key cryptosystems, every protocol problem has a solution.

Furthermore, we present an efficient protocol-generator that on input a (TM)-description of the problem outputs a solution.
Our Result

Assuming existence of public-key cryptosystems, every protocol problem has a solution.

Furthermore, we present an efficient protocol-generator that on input a (TM)-description of the problem outputs a solution.

E.g.

If factoring integers is infeasible then our assumption holds.
Our Protocol-Generator

- (Independent of the problem, )
  outputs a fault-tolerant protocol
  for simultaneous commitment.

- (1) Construct a protocol, for
  "semi-honest" players,
  achieving max. privacy.

(2) Compile this protocol to
  make it fault-tolerant,
  preserving correctness & privacy.
1ST - SIMULTANEOUS COMMITMENT

- A KEY NOTION:

SECRET SHARING (SS)

1. MINORITY OF SHARES YIELDS NOTHING.

2. MAJORITY OF SHARES YIELDS THE SECRET.
1st - Simultaneous Commitment

- A Key Notion:

Verifiable Secret Sharing (VSS)

(1) Minoritiy of shares yields nothing.
(2) Majority of shares yields the secret.
(3) Verifiability of shares!

Implementing VSS = [Shamir] + [GMW]

Simultaneous Commitment is linearly reducible to VSS. [CGMAJ]
1st - Simultaneous Commitment

- A key notion:
  
  **Verifiable Secret Sharing (VSS)**

  - Minority of shares yields nothing.
  - Majority of shares yields the secret.
  - Verifiability of shares!

- Implementing VSS = [Shamir] + [GMW]

- Simultaneous commitment is logarithmically linearly reducible to VSS. [CCMAJ]
2ND - MAX. PRIVACY FOR "SEMI-HONEST"

- **WHAT IS A SEMI-HONEST PARTY?**
  EXECUTES PROTOCOL PROPERLY, BUT RECORDS **ALL** INTERMEDIATE RESULTS.

- **A MAX. PRIVACY PROTOCOL FOR** T
  WHATEVER CAN BE EFFICIENTLY COMPUTED AFTER PARTICIPATING (SEMI-HONESTLY) IN THE PROTOCOL

- **HOW TO CONSTRUCT A MAX. PRIVACY PROTOCOL?**
  ESSENCE OF THIS WORK
Polynomial Indistinguishability

\[ \text{Yao} \quad \text{[GMI]} \]

\[ X_1 \]

\[ T \]

\[ p_1 \equiv \text{Prob}(T(X_1) = 1) \]

\[ X_2 \]

\[ T \]

\[ p_2 \equiv \text{Prob}(T(X_2) = 1) \]

\[ T \quad \text{does not distinguish} \quad \text{if} \]

\[ p_1 = p_2 \]
Polynomial Indistinguishability

\[ \text{Prob}(T(X_1^{(n)}) = 1) = \text{Prob}(T(X_2^{(n)}) = 1) \]

\[ P_1^{(n)} = \text{Prob}(T(X_1^{(n)}) = 1) \]
\[ P_2^{(n)} = \text{Prob}(T(X_2^{(n)}) = 1) \]

T does not distinguish if
Polynomial Indistinguishability

\[ \begin{align*}
X_1^{(m)} & \quad \text{T} \\
T & \quad X_2^{(m)}
\end{align*} \]

\[ P_1^{(m)} = \text{Prob}(T(X_1^{(m)}) = 1) \quad P_2^{(m)} = \text{Prob}(T(X_2^{(m)}) = 1) \]

T does not distinguish if

\[ \forall \epsilon > 0, \; |P_1^{(m)} - P_2^{(m)}| < n^{-c} \quad (n \text{ large enough}) \]

\[ X_1 = 2X_1^{(m)} & \times X_2 = 2X_2^{(m)} \]

are polynomially indistinguishable if \( \forall \text{ Prob. poly-time } T \)

cannot distinguish them.
3rd - forcing semi-honest behaviour

- Idea: "Append" to each message a zero-knowledge proof that it is computed properly.

- A zero-knowledge proof is a "convincing argument" that yields nothing buts the validity of statement.

- Zero-knowledge proofs exist for every NP-statement \[\text{GMW}\], and that's all we need!
3rd - Forcing Semi-Honest Behaviour

• Idea: "Append" to each message a **zero-knowledge proof** that it is computed properly.

• Details (for experts only):

  (1) To deal with **randomized protocols** we use **distributed coin-flipping**, which is implemented using **Sim. Commit**.

  (2) We need and have **auxiliary-input ZK proofs** for every **NP-statement**.
MAX. PRIVACY PROTOCOLS FOR SEMI-HONEST

- IDEA: "DISTRIBUTED SIMULATION" OF BOOLEAN CIRCUIT EVALUATION.

- "SHARING" A PRIVATE INPUT

\[ b = \bigoplus_i b_i = \sum_i b_i \quad \text{GF}(2) \]

- THE "SIMULATION"
DISTRIBUTED SIMULATION OF \textbf{AND} & \textbf{NOT}

\[ b \xrightarrow{\neg} \neg b \]

\[ a \quad b \]

\[ c = a \lor b \]

\[ a_1 \ldots a_n \quad b_1 \ldots b_n \]

\[ c_1 \ldots c_n \]

\[ \text{s.t. } \sum c_i = (\sum a_i) \cdot (\sum b_i) \]
DISTRIBUTED SIMULATION OF AND & NOT

\[ b \quad \rightarrow \quad \Theta \quad \rightarrow \quad b \]

\[ a \quad \rightarrow \quad b \quad \rightarrow \quad c = a \land b \]

\[ a_1 \ldots a_n \quad \rightarrow \quad b_1 \ldots b_n \quad \rightarrow \quad c_1 \ldots c_n \]

\[ \sum c_i = (\sum a_i) \cdot (\sum b_i) \]

\[ \sum c_i = \sum_{i=1}^n a_i \cdot b_i + \sum_{i \neq j}^n (a_i b_j + a_j b_i) \]

\[ c_i^{(i)} \quad \text{easy!} \]

\[ c_i^{(i)} + c_j^{(i)} \quad \text{how?} \]
A max. privacy protocol for $x_1 y_1 + x_2 y_2$

- **What we need**
  
  $X$
  
  $x_0, x_1, x_2$

  $Y$

  $y_1, y_2$

  **Max. privacy protocol**

  $x_0 + x_1 y_1 + x_2 y_2$

- **Reduction to 1-out-of-4 OT**
A MAX. PRIVACY PROTOCOL FOR $x_1 y_1 + x_2 y_2$

- WHAT WE NEED

  $X$

  $x_0, x_1, x_2$

  $Y$

  $y_1, y_2$

  $\text{MAX. PRIVACY PROTOCOL}$

  $x_0 + x_1 y_1 + x_2 y_2$

- REDUCTION TO 1-OUT-OF-2 OT [EGL]

  $S$

  $S_1, S_2$

  $R$

  $i \in \{0, 1, 3\}$

  1-OUT-OF-2 OT

  $s_i$
A Max. Privacy Protocol for $x_1y_1 + x_2y_2$

- **What We Need**

  ![Diagram of X and Y sets]

  - $X = x_0, x_1, x_2$
  - $Y = y_1, y_2$

  **Max. Privacy Protocol**

  - $x_0 + x_1y_1 + x_2y_2$

- **Reduction to 1-out-of-4 OT**

  - $X: x_0, x_1, y_1, y_2$
  - $Y: y_1, y_2$

  ![Truth table and 1-out-of-4 OT diagram]
IMPLEMENTING 1-out-of-2 OT

S

Secret bits $s_1, s_2$

R

Interested in $i \in \{1, 2\}$

Choose $f_T(\cdot) (+ b_T(\cdot))$

\[ \pi_j \leftarrow f_T^{-1}(\pi_j) \]

\[ s_j' \leftarrow s_j \oplus b_T(\pi_j') \]

\[ s_i \leftarrow s_i \oplus b_T(\pi_i) \]

\[ s_{i+1} = s_{i+1} \oplus b_T(\pi_{i+1}) \]

But $s_{i+1}$ remains unpredictable.
Summing Up

**THM:** Every protocol problem has a solution.

Furthermore, a solution can be found efficiently!
A GAME (OF INCOMPLETE INFORMATION)
[von Neumann & Morgenstern]

\( S \)

\( K_2 : S \rightarrow S \)

\( M \)

\( \mu : S_2 \rightarrow M \)

\( \delta : S \times M \rightarrow S \)

\( g : S \rightarrow V \)

STATES OF THE GAME

KNOWLEDGE FUNCTIONS

possible MOVES

STRATEGIES

TRANSITION FUNCTION

PAYOFF FUNCTION
A GAME (OF INCOMPLETE INFORMATION)

[S von Neumann & Morgenstern]

$S$  STATES OF THE GAME
$K_i : S \rightarrow S_i$  KNOWLEDGE FUNCTIONS
$M$  possible MOVES
$\mu_i : S_i \rightarrow M$  STRATEGIES
$\delta : S \times M \rightarrow S$  TRANSITION FUNCTION
$\pi : S \rightarrow V$  PAYOFF FUNCTION

- How to select an optimal STRATEGY?
A GAME (OF INCOMPLETE INFORMATION)  
[von Neumann & Morgenstern]

\[ S \]  \hspace{2cm} \text{STATES OF THE GAME}

\[ K_i : S \to S_i \]  \hspace{2cm} \text{COMPUT' KNOWLEDGE FUNCTIONS}

\[ M \]  \hspace{2cm} \text{POSSIBLE MOVES}

\[ \mu_i : S_i \to M \]  \hspace{2cm} \text{STRATEGIES}

\[ \delta : S \times M \to S \]  \hspace{2cm} \text{COMPUT' TRANSITION FUNCTION}

\[ \phi : S \to V \]  \hspace{2cm} \text{COMPUT' PAYOFF FUNCTION}

- How to implement a game?

Special cases:

1. A Protocol Problem
2. A Generalized Protocol Problem (with on-line ext. inputs)
   - Initial state = \( \emptyset \)
   - Moves = Inputs
IMPLEMENTING A COMPUTABLE GAME

- COMMIT TO A MOVE
- DISTRIBUTIVELY COMPUTE THE NEXT STATE.
- DISTRIBUTIVELY COMPUTE PARTIAL INFORMATION (FOR EACH PLAYER!)
- DISTRIBUTIVELY COMPUTE THE FINAL PAYOFF.

Repeat (as long as the game continued)
Summary

- General Results (obtained)
  - How to solve any protocol problem.
  - How to play any game.

- The cost (to be reduced)
  - Passing to circuit model
    - PRAM \(\rightarrow\) Circuit (TM \(\rightarrow\) Circuit)
    - Work with bounded degree network?
  - \(\Theta(n^2)\) communication per each elementary step.
    - Another model?
    - Amortize?
  - ZK proofs on each message
    - Postpone proofs to the end, get rid of OT.