Efficient and Secure Multi-Party Computation with Faulty Majority and Complete Fairness

Juan Garay (Bell Labs)
Philip MacKenzie (Bell Labs)
Ke Yang (CMU)

Multi-Party Computation (MPC)

Parties $P_1, P_2, ..., P_n$ (some corrupted), each holding private input x_i, wish to compute $y = f(x_1, x_2, ..., x_n)$ privately and correctly.

Security is normally formulated in a simulation paradigm —

- Real world: parties carry out the protocol.
  Adversary $\mathcal{A}$ controls communication and corrupts parties.
- Ideal process: a functionality $\mathcal{F}_f$ does the computation.
  Ideal adversary $\mathcal{S}$ is rather limited.
- Security: Π securely realizes $\mathcal{F}_f$, if
  $$\forall \mathcal{A}\ \exists \mathcal{S} \text{ s.t. } \textsc{View}(\Pi, \mathcal{A}) \approx \textsc{View}(\mathcal{F}_f, \mathcal{S}).$$

Variants of Security Definitions

First in [Goldreich-Micali-Wigderson ’87], de facto standard w/ variants...

- synchronous/asynchronous communication
- stand-alone/concurrent
- single-invocation/reactive

[GL91, MR91, B91, C00, PW00, PW01...]

Universally Composable (UC) framework [Canetti ’01] — asynchronous network, reactive, allows arbitrary composition (very strong security).

Fair Multi-Party Computation (FMPC)

Parties $P_1, P_2, ..., P_n$ (some corrupted), each having private input x_i, wish to compute $y = f(x_1, x_2, ..., x_n)$ privately and correctly.

Security is about absolute information gain.
"The only information each party learns is at most y."

Fairness is about relative information gain.
"Either everyone learns y, or no one learns anything."
(also known as complete fairness)

Important in MPC, crucial in some applications.
(e.g. two-party contract-signing)

Fairness: Positive Results

n parties, t corrupted.

- $t < n/3$ — possible with a point-to-point network.
  - computational [Goldreich-Micali-Wigderson '87]
  - information theoretical [Ben-Or Goldwasser Wigderson '88, Chaum-Crépeau-Damgård '88]
- $n/3 \leq t < n/2$ — possible with a broadcast network.
  - computational [Goldreich-Micali-Wigderson '87]
  - information theoretical [Rabin Ben-Or '89]

Unfortunately...

Fairness is impossible with corrupted majority $(t \geq n/2)$...

[Cleve '86] No fair two-party coin-tossing protocol exists.

Intuition: The party sending the last message may abort early...

Formally: for every protocol Π, there exists an adversary $\mathcal{A}$, s.t.
$\mathcal{A}$ gains unfair advantage in Π by aborting.

Consequently, many security definitions do not consider fairness or only consider partial fairness.

[BG90, GL91, FGHHS02, GL02...]

In particular, the UC framework is completely unfair.

Fairness After the Impossility Result

We still need fairness, so tweak the model/definition...

“Optimistic” approach (tweak the model) [ASW98, CC00...]

- Adds a trusted party as an arbitor in case of dispute.
- Needs to be (constantly) “available.”

“Gradual Release” approach (tweak the definition) [B83, D95...]

- Parties take turn to release information “little-by-little.”
- Still somewhat unfair, but can control the “unfairness.”

The Gradual Release Approach

We stay in the standard model, tweak the definition...

Problems:

1. How to tweak the security definition?
   minimal modification, within the simulation paradigm...

2. How to construct fair protocols?
   simplicity, reuse the existing (unfair) MPC protocols, efficiency...

3. How robust are these protocols?
   concurrency, non-malleability, security after composition...

Our Contributions

1. a “minimally tweaked” definition

   The first one that is completely in the simulation paradigm.

2. efficient constructions of fair MPC protocols

   A general technique to convert unfair protocols into fair ones efficiently.

3. robust security

   Preserves security under arbitrary composition.

Agenda

1. a “minimally tweaked” definition

   The first one that is completely in the simulation paradigm.

2. efficient constructions of fair MPC protocols

   A general technique to convert unfair protocols into fair ones efficiently.

3. robust security

   Preserves security under arbitrary composition.

Previous (*ad hoc*) Security Definitions

A typical gradual release protocol [BN00, P03] consists of two phases.

computing phase: "normal" computation

revealing phase: each P_i gradually reveals a "secret" s_i;
then each P_i computes the result y from $s_1, s_2, ..., s_n$.

(*Ad hoc*) security definition:

1. The computation phase is simulatable.
2. The revealing phase is simulatable if $\mathcal{S}$ knows y.
3. If $\mathcal{A}$ can find y in time t, then honest parties can find y in time "comparable" to t.

Problems with Previous Definitions

Security definition:

1. The computation phase is simulatable.
2. The revealing phase is simulatable if $\mathcal{S}$ knows y.
3. If $\mathcal{A}$ can find y in time t, then honest parties can find y in time "comparable" to t.

The definition is not in the simulation paradigm...

Suppose $\mathcal{A}$ aborts early and doesn't have enough time to find y.
Then $\mathcal{S}$ doesn't know y either...
But then the revealing phase is not simulatable!

Consequently: $\mathcal{A}$ may gain unfair advantage by simply aborting early!

This becomes even worse when we compose protocols...

Our Security Definition ("Quantifier Switch")

- [Cleve '86] $\forall \Pi\ \exists \mathcal{A}$ s.t. $\mathcal{A}$ gains unfair advantage in Π.
- This work: allows Π to depend on the running time of $\mathcal{A}$...
- Timed protocol $\Pi[T] = \{\Pi[t]\}$: parameterized by the adversary's running time.
- Security definition: $\Pi[T]$ securely realizes $\mathcal{F}_f$, if there exists a black-box ideal adversary $\mathcal{S}$, s.t. for all t, for any adversary $\mathcal{A}$ of time t,

$$\textsc{View}(\Pi[t], \mathcal{A}) \approx \textsc{View}(\mathcal{F}_f, \mathcal{S}^{\mathcal{A}})$$

for any distinguisher (environment) of running time t.

What About Fairness?

Fairness Definition: A timed protocol $\Pi[T]$ is λ-fair, if each honest party's running time in $\Pi[t]$ is bounded by $\lambda \cdot t + p$ for a fixed polynomial p.

- $\Pi[T]$ is fair, if $\lambda = O(n)$.

Strange? honest parties may run longer than corrupted ones...

- By aborting, corrupted parties may gain unfair advantage, and honest parties need more time to catch up...
- Worst case: 1 honest party against $n - 1$ corrupted ones — needs $O(n)\times$ more time.
- Reasonable: in normal executions, the running time of honest parties are independent of t.

Our Definitions

- security definition:
  - a (minimal) “quantifier switch” tweak
  - avoids the impossibility result
  - completely in the simulation paradigm
- fairness definition:

  separated from security definition, entirely based on the running time

Agenda

1. a “minimally tweaked” definition

   The first one that is completely in the simulation paradigm.

2. efficient constructions of fair MPC protocols

   A general technique to convert unfair protocols into fair ones efficiently.

3. robust security

   Preserves security under arbitrary composition.

Observation on Existing MPC Protocols

Many (unfair) MPC protocols share the same structure:
(e.g. [Goldreich-Micali-Wigderson ’87], [Cramer-Damgård-Nielsen ’01], [Canetti-Lindell-Ostrovsky-Sahai ’02]...)

sharing phase parties share data among themselves
(simple sharing, or $(n, n-1)$ threshold sharing)

evaluation phase “gate-by-gate” evaluation
(all the intermediate data are shared or “blinded”)

revelation phase each party reveals its secret share
(all parties then learn the result from the shares)

What Make These Protocols Unfair?

The protocols are unfair because of the revelation phase.

A worst-case scenario:

1. Honest parties reveal their secrets.
2. Corrupted parties abort and learn the result.
3. Honest parties learn nothing.

If we make the revelation phase fair, the entire protocol is fair!

$\mathcal{F}_{\text{CPFO}}$: Commit-Prove-Fair-Open

- commit phase: every party P_i commits to a value x_i.
- prove phase: every party P_i proves a relation about x_i.
- open phase: open $x_1, x_2, ..., x_n$ simultaneously.

Simultaneous opening guarantees fairness — either all parties learns all the committed values, or no one learns anything.

Using $\mathcal{F}_{\text{CPFO}}$, the revelation phase becomes fair, so does the protocol...

Time-Lines: Towards Realizing CPFO

- $N = p_1 p_2$ is a Blum integer; g is a random element in $\mathbb{Z}_N^*$.
- $\vec{G}$ is a vector $\vec{G} = (g, g^2, g^{2^2}, ..., g^{2^{2^k}})$, or $G[i] = g^{2^i}$.

Observations [Boneh-Naor 00, Garay-Jakbsson 02]:

- Sequential access: can move forward ℓ positions by doing ℓ squarings.
- Random access: knowing $\phi(N)$, one can compute $G[i]$ for any i.
- Conjecture: not knowing the factorization of N, hard to move backward, and inefficient to move forward (needs to do step-by-step).
- So a point “far away” is “unknown”...

Assumptions

- $N = p_1 p_2$ is a Blum integer; g is a random element in $\mathbb{Z}_N^*$.
- $\vec{G}$ is a vector $\vec{G} = (g, g^2, g^{2^2}, ..., g^{2^{2^k}})$, or $G[i] = g^{2^i}$.

hard to move backward, inefficient to move forward (sequential access)...

"generalized BBS" [Boneh-Naor '00]

Given $(G[2^1], G[2^2], ..., G[2^\ell])$, for an adversary of running time $\delta \cdot 2^\ell$, $G[2^{\ell+1}]$ still appears pseudorandom.

"yet-more-general BBS (YMG-BBS)" [This work]

Suppose $|a_{\ell+1} - a_i| \geq 2^\ell$ for $i = 1, 2, ..., \ell$.
Given $(G[a_1], G[a_2], ..., G[a_\ell])$, for an adversary of running time $\delta \cdot 2^\ell$, $G[a_{\ell+1}]$ still appears pseudorandom.

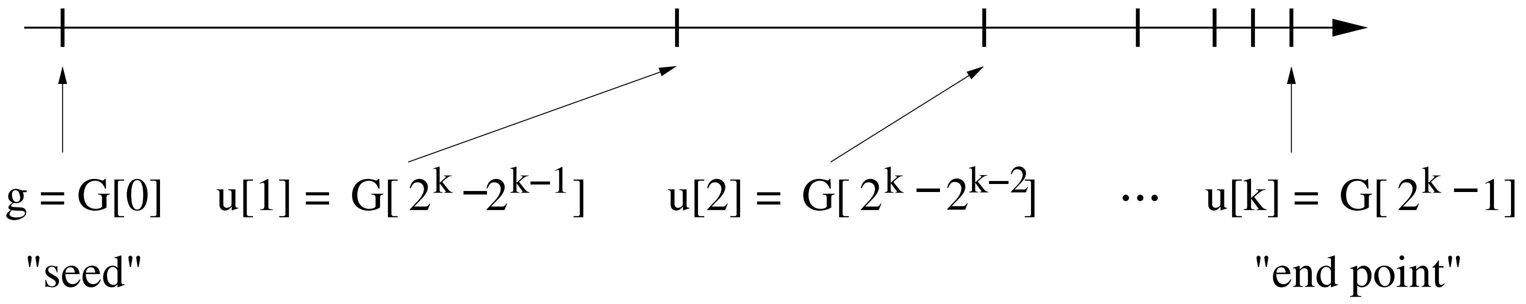
YMG-BBS is also (implicitly) used in [P03,GP03].

Time-Lines

- $N = p_1 p_2$ is a Blum integer; g is a random element in $\mathbb{Z}_N^*$.
- $\vec{G}$ is a vector $\vec{G} = (g, g^2, g^{2^2}, ..., g^{2^{2^k}})$, or $G[i] = g^{2^i}$.

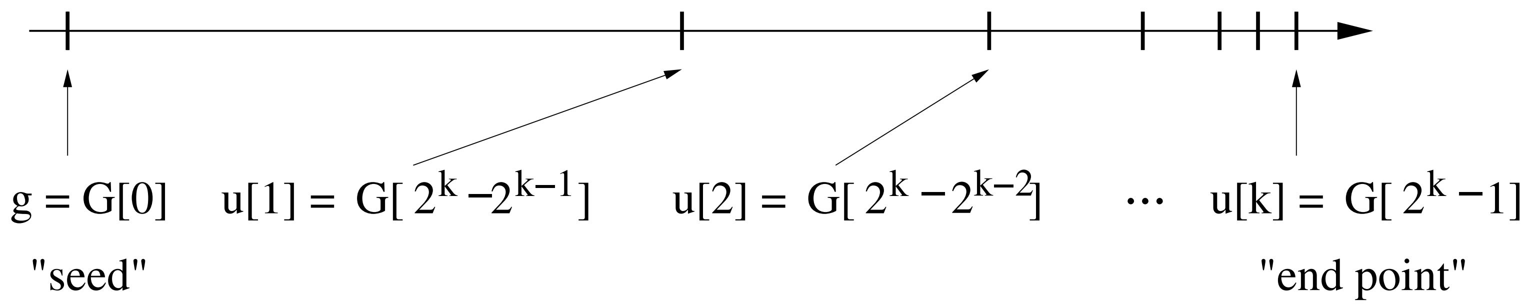
(decreasing) time-line: $T = \langle N, g, \vec{u} \rangle$, $u[i] = G[2^k - 2^{k-i}], \quad i = 1, ..., k$.

$g = G[0]$ is the seed; $u[k] = G[2^k - 1]$ is the end point.



"ascending points with exponentially descreasing steps"

Time-Line Commitments

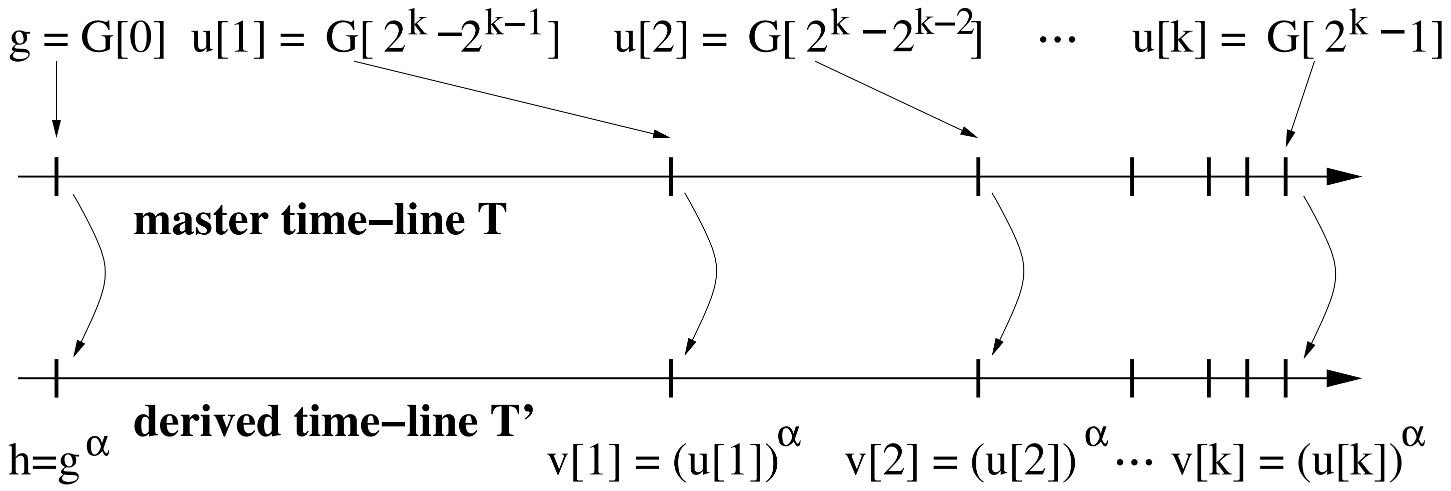


- The seed $g = G[0]$ determines the time-line.
- The end point $u[k] = G[2^k - 1]$ is $(2^k - 1)$ “squarings” away from g and appears pseudorandom.
- A time-line commitment to x is $(g, x \cdot u[k])$: takes $(2^k - 1)$ squarings to force-open.
- Gradual release: reveals $u[1], u[2], ..., u[k]$ one by one, each cuts the number of squarings by half.

Derived Time-Lines [GJ02]

master time-line $T = \langle N, g, \vec{u} \rangle$

derived time-line $T' = \langle N, h, \vec{v} \rangle$: $h = g^{\alpha}$, $v[i] = (u[i])^{\alpha}$.



Strong Pseudorandomness: $v[k]$ appears pseudorandom even if the entire master time-line T is known, but the shifting factor α is unknown.

Realizing CPFO Using Time-Lines

Set-up: a master time-line $T = \langle N, g, \vec{u} \rangle$ in CRS.

Commit: for each party P_i:

derives a time-line $T_i = \langle N, g_i, \vec{u_i} \rangle$;

time-line commits to x_i using $(g_i, x_i \cdot u_i[k])$.

Prove: standard ZK proof.

Open: in round ℓ, each party P_i reveals $u_i[\ell]$ with a ZK proof;

if any party aborts, enter panic mode.

Panic mode: depends on the current round ℓ...

- If $(k - \ell)$ is large, then abort.
  ($\mathcal{A}$ does not have enough time to force-open)
- If $(k - \ell)$ is small, then force-open.
  ($\mathcal{A}$ has enough time to force-open as well)

Putting Things Together...

Plug $\mathcal{F}_{\mathsf{CPFO}}$ into existing MPC protocols $\longrightarrow$ Fair MPC protocols

- [Canetti-Lindell-Ostrovsky-Sahai ’02]
  a fair MPC protocol in the CRS model
- [Cramer-Damgård-Nielsen ’01]
  an efficient fair MPC protocol in the PKI model
  - the CDN protocol is efficient
  - added $\mathcal{F}_{\mathsf{CPFO}}$ can be realized efficiently

Agenda

1. a “minimally tweaked” definition

   The first one that is completely in the simulation paradigm.

2. efficient constructions of fair MPC protocols

   A general technique to convert unfair protocols into fair ones efficiently.

3. robust security

   Preserves security under arbitrary composition.

A Fair MPC Framework

Fair MPC (FMPC): variant of the UC framework to make fairness possible.

- Real world/ideal process: synchronous broadcast with rounds
  Asynchronous communication is inherently unfair (e.g. starvation).
- Ideal process: results from ideal functionality go directly to parties
  In UC, $\mathcal{S}$ may not forward the results, making the protocol unfair.

Caution: FMPC only provides the possibility to have fair ideal functionalities and fair protocols, not a guarantee; it is always possible to have unfair functionalities/protocols.

A Composition Theorem in FMPC Framework

Similar to the composition theorem in UC...

- More complicated since we deal with timed protocols.
  needs to consider the precise running time of adversaries...

- Intuitively, any secure protocol in FMPC remains secure when arbitrarily composed.
  in particular, they are concurrently secure and non-malleable...

Our Constructions in FMPC Framework

- [Canetti-Lindell-Ostrovsky-Sahai ’02] (fair MPC in CRS model)
  CLOS secure in UC: converted protocol is secure in FMPC
- [Cramer-Damgård-Nielsen ’01] (efficient fair MPC in PKI model)
  CDN only secure in modular composition model
  - CDN use a plain trapdoor commitment in their ZK proofs.
  - We change it to a simulation sound trapdoor commitment (SSTC);
  - Add encryption of the witness (admits non-rewinding extractors);
  - Makes the ZK universally composable.
  - [Damgård-Nielsen ’03] uses a universally composable commitment instead of an SSTC, but SSTCs are more efficient.
  - The converted and strengthened CDN is fair and secure in FMPC.

An Example: Socialist Millionaires' Problem

Millionaire's problem: $f(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 \geq x_2 \\ 0 & \text{if } x_1 < x_2 \end{cases}$

[Yao '82, Yao '86] no known efficient solutions

Socialist millionaire's problem: $f(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{if } x_1 = x_2 \end{cases}$

[Fagin-Naor-Winkler '96] crytographic and non-cryptographic solutions

[Jakobsson-Yung '96, Naor-Pinkas '99] efficient (unfair) solutions

[Boudot-Schoenmaker-Traoré '01] efficient fair solutions w/ *ad hoc* security

[This work] efficient, fair solution in FMPC framework

Approximating Socialist Millionaires' Problem

SMP can be approximated by $g_r(x_1, x_2) = r \cdot (x_1 - x_2) \bmod N$.

- r a random element in $\mathbb{Z}_N^*$.
- if $x_1 = x_2$, then $g_r(x_1, x_2) = 0$.
- if $x_1 \neq x_2$ and $GCD(x_1 - x_2, N) = 1$, then $g_r(x_1, x_2)$ is a random element in $\mathbb{Z}_N^*$.
- Cramer-Damgård-Nielsen approach: very efficient protocol (3 gates)
- can enhance security by changing the ZK proofs in CDN to UCZK
- plugging in $\mathcal{F}_{\text{CPFO}}$ makes it fair

Putting things together: an efficient, fair, and secure protocol for SMP.

Summary

- fairness in MPC, impossibility result for corrupted majority
- tweaked definition (timed protocols, quantifier switch)
- $\mathcal{F}_{\mathsf{CPFO}}$: converting unfair MPC protocols into fair ones.
- realizing $\mathcal{F}_{\mathsf{CPFO}}$ efficiently using time-lines
- fair MPC framework (like UC) — robust security
- example — socialist millionaires' problem

Thank You!

- Juan Garay garay@research.bell-labs.com
- Philip MacKenzie philmac@lucent.com
- Ke Yang yangke@cs.cmu.edu

full paper available at IACR eprint, report 2004/009