Secure Multiparty Computation: An Introduction

Yuval Ishai

Technion
Going digital...
How to store important data
How to store important data
How to store important data
How to store important data

Data is virtually indestructible

Error-correcting codes: Hamming 1947, Shannon 1948, …

Similar level of integrity with far less storage
What about confidentiality?

Single point of failure!

Single point of failure!

Single point of failure!
What about confidentiality?
Use Secret Sharing!

Data is virtually unleakable AND indestructible…
Can we still search the data?
Can we still search the data?

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$X_6$</th>
<th>$X_7$</th>
</tr>
</thead>
</table>

![Cloud Storage logos](image-url)
Can we still search the data?

Single point of failure!
Can we still search the data?

Single point of failure!
Can we still search the data?

Single point of failure!

Find expenses > $1000
Can we still search the data?

Single point of failure!

Should I see a doctor?
Should I see a doctor?

No.
A Decentralized Alternative

Find alarming health trends.
A Decentralized Alternative

Alibaba Cloud  box  Dropbox  iCloud  Google Drive  OneDrive  Yandex.Disk

a₁  a₂  a₃

b₁  b₂  b₃

C₁  C₂  C₃

a  b  c
A Decentralized Alternative
A Decentralized Alternative

Alibaba Cloud  box  Dropbox  iCloud  Google Drive  OneDrive  Yandex.Disk

\[
\begin{align*}
& a_1 & a_2 & a_3 \\
& b_1 & b_2 & b_3 \\
& c_1 & c_2 & c_3
\end{align*}
\]
Secure Multiparty Computation (MPC):

Process sensitive data without introducing a single point of failure
<table>
<thead>
<tr>
<th>confidentiality</th>
<th>Information</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrity</td>
<td>Error-correcting code</td>
<td>Fault-tolerant computation</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Secret-sharing scheme</td>
<td>Secure multiparty computation</td>
</tr>
</tbody>
</table>

+integrity
MPC is more general than it may seem

• Can capture problems from many areas
  – Error-correcting codes
  – Distributed algorithms
  – Interactive proofs, PCPs, randomness extractors
  – Encryption, signatures, zero-knowledge proofs
  – Cryptographic obfuscation, functional encryption
  – Anything that involves “good guys” trying to achieve a common goal in the presence of “bad guys”

• Too big to fail…

• Focus of this talk: secure function evaluation
Rest of Talk

• MPC example
• Defining MPC
• Overview of results
How much do we earn?

Goal: compute $\sum x_i$ without revealing anything else
A better way?

Assumption: $\sum x_i < M$ (say, $M=10^{10}$) (+ and – operations carried out modulo $M$)
A security concern

\[ m_2 = m_1 + x_2 \]
Resisting collusions

\[ x_i + \text{inbox}_i - \text{outbox}_i \]
More generally

- Parties $P_1, \ldots, P_n$ want to compute $f(x_1, \ldots, x_n)$
  - Up to $t$ parties can collude
  - Should learn (essentially) nothing but the output

- Questions
  - When is this at all possible?
  - How efficiently?

Seminal feasibility results from the 1980s:

- Information-theoretic (unconditional) security possible when $t < n/2$
  [BenOr-Goldwasser-Wigderson88, Chaum-Crépeau-Damgård88, Rabin-BenOr89]

- Computational security possible when $t < n$ (under standard assumptions)
  [Yao86, Goldreich-Micali-Wigderson87]
More generally

- Parties $P_1, \ldots, P_n$ want to compute $f(x_1, \ldots, x_n)$
  - Up to $t$ parties can collude
  - Should learn (essentially) nothing but the output

- **Questions**
  - When is this at all possible?
  - How efficiently?

- Several efficiency measures:
  - communication, rounds, computation

- Very active area of research, both theoretical and applied

- Relatively small gap between “provable” and “heuristic” security
Many applications
(some unexpected…)

- Voting, trading, bidding, matching, key management, smart contracts,…
- Private Circuits [ISW03,…]
- MPC => ZK => post-quantum signatures [IKOS07,…,AHIV17,CDG+17,KRW18,…]
- Defeating hardware trojans [DFS16,BGILT18,…]
From Theory to Practice?

https://www.multipartycomputation.com/mpc-software

Security

Google takes the PIS out of advertising: New algo securely analyzes shared encrypted data sets without leaking contents

Plus: MongoDB crams end-to-end crypto into database tech

By Thomas Claburn in San Francisco 19 Jun 2019 at 21:47 11  SHARE
From Theory to Practice?

• Much more room for efficiency improvements
  – Both for general MPC and for useful instances
  – Ideally: approach efficiency of insecure computation
  – Quite far, but no barriers in sight

• Rest of talk: definitions, protocols, recent progress
Definitions
Real/Ideal Paradigm

[Goldwasser-Micali82, Goldwasser-Micali-Rackoff85, Goldreich-Micali-Wigderson87, ..., Canetti01, ...]

• “Whatever an adversary can achieve by attacking the real protocol, it could have also achieved by attacking an ideal protocol that employs a trusted party.”

• Achieve = learn + influence

• Formalized via a simulator

• Captures secrecy, correctness, independence of inputs, ...
Real/Ideal Paradigm

Real protocol

- Adversary
- Honest parties

Ideal protocol

- Trusted party computing $f$
- Simulator
- Honest parties

$x^2 > 7$
Real/Ideal Paradigm

Real protocol

Adversary

Honest parties

Environment Z

0/1

Ideal protocol

Trusted party computing f

Simulator

Honest parties

Environment Z

0/1
Protocol $\pi$ securely realizes $f$ if:

For every $A$ there is $S$ such that for every $Z$,

$$\Pr[\text{Real}(Z,A,\pi)=1] \approx \Pr[\text{Ideal}(Z,S,f)=1]$$

**Standalone MPC**: $Z$ only sends inputs and receives outputs

**Universally Composable MPC**: $Z$ arbitrarily interacts with $A/S$
Real/Ideal Paradigm

Standalone security with “straight-line simulation” + mild technical requirement
→ UC security
[Kushilevitz-Lindell-Rabin10]

Applies to natural protocols in the information-theoretic setting

Standalone MPC: Z only sends inputs and receives outputs
Universally Composable MPC: Z arbitrarily interacts with A/S
Protocol $\pi$ securely realizes $f$ if:
For every $A$ there is $S$ such that for every $Z$,
$$\Pr[\text{Real}(Z, A, \pi) = 1] \cong \Pr[\text{Ideal}(Z, S, f) = 1]$$

Environment $Z$ cannot distinguish between
$\text{REAL}_{A,\pi} = (\text{Output of } A, \text{Output of } H)$ in Real protocol attacked by $A$
$\text{IDEAL}_{S,f} = (\text{Output of } S, \text{Output of } H)$ in Ideal protocol attacked by $S$
Protocol $\pi$ securely realizes $f$ if:
For every $A$ there is $S$ such that for every $Z$,
$$\Pr[\text{Real}(Z, A, \pi) = 1] \cong \Pr[\text{Ideal}(Z, S, f) = 1]$$

Environment $Z$ cannot distinguish between
$\text{REAL}_{A,\pi} = (\text{View of A, Output of H})$ in Real protocol attacked by A
$\text{IDEAL}_{S,f} = (\text{Output of S, Output of H})$ in Ideal protocol attacked by S
Landscape of Definitions

• Many different models… but:
  – answers to most natural questions are only sensitive to very few aspects of model
  – general connections between models
  – few “standard” models

• Defining an MPC task involves specifying
  – **Functionality**: what do we want to achieve?
  – **Network model**: how are we going to do this?
  – **Adversary**: who do we need to protect against?
  – **Security type**: which kind of protection do we want?
Many different models have been proposed, but:
- answers to most natural questions are only sensitive to very few aspects of model
- general connections between models
- few “standard” models

Defining an MPC task involves specifying:
- **Functionality**: what do we want to achieve?
- **Network model**: how are we going to do this?
- **Adversary**: who do we need to protect against?
- **Security type**: which kind of protection do we want?

**Landscape of Definitions**
- Captures the ideal goal
  - Specifies solution using help of trusted party
  - Defines inevitable vulnerabilities
- Variants
  - Deterministic vs. randomized
  - Single output vs. multi-output
  - Non-reactive vs. reactive
Landscape of Definitions

• Many different models… but:

Which functionalities are “safe” to compute?

Out of scope for MPC

Theme of mechanism design, differential privacy, algorithmic fairness

• Defining an MPC task involves specifying
  – Functionality: what do we want to achieve?
  – Network model: how are we going to do this?
  – Adversary: who do we need to protect against?
  – Security type: which kind of protection do we want?
Many different models, but:
- Answers to most natural questions are only sensitive to very few aspects of model
- General connections between models
- Few "standard" models

Defining an MPC task involves specifying:
- Functionality: what do we want to achieve?
- Network model: how are we going to do this?
- Adversary: who do we need to protect against?
- Security type: which kind of protection do we want?

Landscape of Definitions

Synchronous vs. Asynchronous

Secure vs. Insecure channels

Helper functionalities: broadcast, oblivious transfer (OT), ...
Many different models… but:

- answers to most natural questions are only sensitive to very few aspects of model
- general connections between models
- few “standard” models

Defining an MPC task involves specifying:

- **Functionality**: what do we want to achieve?
- **Network model**: how are we going to do this?
- **Adversary**: who do we need to protect against?
- **Security type**: which kind of protection do we want?

Landscape of Definitions

Possible sets of corrupted parties
- Typically a threshold \( t \)

Passive (semi-honest) vs. Active (malicious)

Computationally bounded vs. Unbounded

Static vs. adaptive vs. mobile

- Functionality: what do we want to achieve?
- Network model: how are we going to do this?
- Adversary: who do we need to protect against?
- Security type: which kind of protection do we want?
Many different models… but:

- answers to most natural questions are only sensitive to very few aspects of model

**Landscape of Definitions**

- Standalone vs. Universally Composable (UC)
- Perfect vs. Statistical vs. Computational
- Full security vs. Security with abort

[McCoy86]: Generally impossible unless \( t < n/2 \)

In ideal protocol: First S gets its outputs, then decides if to abort
More explicitly...

- **Simplest setting:**
  - Perfect security over secure channels
  - Deterministic, single-output $f$
  - Passive adversary
  - Unbounded simulator

- $\pi$ is a $t$-secure protocol for $f$ if:
  - It correctly computes $f$
  - $\forall T \subseteq [n]$ of size $\leq t$, $\forall x, x'$ such that $x_T = x'_T$ and $f(x) = f(x')$

  $\text{View}_T(x) \equiv \text{View}_T(x')$
More explicitly...

- **Simplest setting:**
  - **Perfect** security over secure channels
  - **Deterministic, single-output** $f$
  - **Passive** adversary
  - **Unbounded** simulator

- $\pi$ is a $t$-secure protocol for $f$ if:
  - It correctly computes $f$
  - $\forall T \subseteq [n]$ of size $\leq t$, $\forall x,x'$ such that $x_T = x'_T$ and $f(x) = f(x')$
    $$\text{View}_T(x) \equiv \text{View}_T(x')$$

- **Q:** For which $f$ does such $\pi$ exist?
  - All $f$ when $t < n/2$ [BGW88, CCD88, ...]
  - Open for bigger $t$, except when $n = 2$ [CK89, Kus89, Bea89, ...]
How do general protocols work?
4 Approaches to MPC

**Garbled Circuits**
[Yao 86,...]

**Fully Homomorphic Encryption**
[Gentry 09,...]

**Linear Secret Sharing**
[Goldreich-Micali-Wigderson 87]
[BenOr-Goldwasser-W88, Chaum-Crépeau-Damgård88, ...]

**Homomorphic Secret Sharing**
[Boyle-Gilboa-I 15,...]
In hardware we trust?
In hardware we trust?
In hardware we trust?

Potential synergy with MPC:
- Build trusted hardware using simple MPC
- MPC on top of trusted hardware for best-of-both-worlds security
The TPMPC workshops aim to bring together practitioners and theorists working in multi-party computation. This year's event will be held in Aarhus, Denmark from May 25th to May 28th.

### Call for Contributed Talks ###

Deadline: 25 February 2020

TPMPC solicits contributed talks in the area of the theory and/or practice of secure multiparty computation. Talks can include papers published recently in top conferences, or work yet to be published. Areas of interest include:

- Theoretical foundations of multiparty computation: feasibility, assumptions, asymptotic efficiency, etc.
- Efficient MPC protocols for general or specific tasks of interest
- Implementations and applications of MPC

For further details regarding contributed talks and submissions, see: [https://www.multipartycomputation.com/tpmpc-2020](https://www.multipartycomputation.com/tpmpc-2020)