Efficient and Fair MPC using Blockchain and Trusted Hardware

Souradyuti Paul
(IIT Bhilai)

Ananya Shrivastava
(IIT Gandhinagar)

Latincrypt 2019 Santiago, Chile
October 3, 2019
Outline

- Multiparty Computation (MPC)
  - Security Property of MPC: Privacy, Correctness, Fairness
- Various Components
  - Blockchain
  - Trusted Hardware
  - Core MPC having privacy and correctness security
- Fair MPC Protocol using Blockchain and Trusted Hardware: CGJ+ Protocol
- Attack on CGJ+ Protocol
- Our Construction
- Results
Multiparty Computation (MPC)

Definition (Informal)

There are $n$ parties $P_1, P_2, \ldots, P_n$ who do not trust each other. Each party $P_i$ has its own private input $x_i$ and there is a common function $f(.)$ with $n$-bit input that every party wants to compute on their private data.

$$y = f(x_1, x_2, x_3, x_4) = (x_1 + x_2 + x_3 + x_4)$$
Security Property of MPC: Fairness

**Definition (Informal)**

An adversary can receive their output only if all honest parties receive output.
Component 1: Bulletin Board (Blockchain)

Properties:

- Messages are permanently available.
- Messages are visible publicly to all the parties.
- Produces a publicly verifiable proof that the message is posted publicly.
- Generates proofs using an Authentication Scheme which can be publicly verified.

Public Ledger BB
Component 2: Trusted Hardware

Properties:

- It provides the private regions of memory -- known as enclaves -- for running programs.
- An enclave provides confidentiality and integrity of a program in the presence of adversarial environment.
- It provides attestation of the correct execution of a program using digital signatures.
- Example: Intel Software Guard Extension (SGX)
Component 3: Core MPC having *privacy* and *correctness* security

Here, $ct = AE.Enc((k_0, k_1), f(x, y))$
Generic Structure of the Protocol
Fair MPC Protocol using BB and Trusted Hardware: CGJ+ Protocol

Secrets: $x$  

Compute: $f(x,y)$

---

CGJ+ Protocol

Stage 1

Stage 2

Stage 3
CGJ+ Protocol: Stage 1

\[ k_0, \rho_0, r_0, t_0 \quad com_0 = \text{Com}(k_0, r_0) \quad ct_0 = \text{AE.Enc}_{DHK}(k_0, \rho_0, t_0) \quad k_1, \rho_1, r_1, t_1 \]

\[ com_1 = \text{Com}(k_1, r_1) \quad ct_1 = \text{AE.Enc}_{DHK}(k_1, \rho_1, t_1) \]
CGJ+ Protocol: Stage 2
CGJ+ Protocol: Stage 3
Our Observation

- The security of CGJ+ protocol is proved (in the malicious model with dishonest majority) under the condition that the core MPC component π supports the privacy of the individual secrets, and the correctness of the output.
- While privacy is ensured using a secret-sharing scheme, achieving correctness of output requires expensive operations such as ZKP and commitment schemes.

*Can we break the fairness property of the CGJ+ protocol, if the core MPC component π is allowed to output an incorrect value?*
Fairness Attack on CGJ+ Protocol

Stage 2

\[ x, k_0, \text{com}_0 \rightarrow \text{Protocol } \pi \rightarrow y, k_1, \text{com}_1 \]

Stage 3

\[ \text{Public Ledger BB} \]

\[ \text{Enclave } \mathcal{G} \]

\[ \text{Partition 0, Partition 1} \]

UNFAIR
Our Construction

- Designed a new fair protocol $\Gamma$, which works even if the internal component $\pi$ returns an incorrect value.
- We reiterate that the origin of the attack in CGJ+ protocol is the *release tokens* $(\rho_0, \rho_1)$ being generated independently of the ciphertext.
- We remove the *release tokens* altogether from the protocol and generate a tag from BB using the ciphertext directly.
Our Construction: Stage 1
Our Construction: Stage 2

Protocol \( \pi \)

\[ x, k_0 \rightarrow \text{ct} \rightarrow \pi \rightarrow y, k_1 \rightarrow \text{ct} \]
Our Construction: Stage 3
Summary of Our Contribution

- Our first contribution is showing concrete fairness attacks on the protocols described in CGJ+ (denoted by \( \Pi \)) and KMG\(^2 \) (stateless version of CGJ+) protocols, when the underlying protocol \( \pi \) allows incorrect output to be returned.
- Next, we design a new protocol \( \Gamma \) based on public ledger and trusted hardware, and prove that it is fair, even if \( \pi \) returns an incorrect value.
- We extended our work to design a stateless version of \( \Gamma \), namely \( \Upsilon \), and also prove its fairness.

## Results

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Stateful/Stateless</th>
<th>Primitives used in $\pi$</th>
<th>$\pi$ security Def. 3</th>
<th>$\pi$ security Def. 4</th>
<th>$O(k + \lambda)$</th>
<th>$O(k + \lambda)$ bits</th>
<th># of var. in $\mathcal{G}$</th>
<th># of calls in $\mathcal{G}$</th>
<th>Comm.:</th>
<th>Enc.:</th>
<th>Dec.:</th>
<th>OWF:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi$</td>
<td>Stateful</td>
<td>SSS + AE + MAC + ZKPoPK</td>
<td>Fair</td>
<td>Attack</td>
<td>0 bits</td>
<td>$O(k + \lambda)$ bits</td>
<td>13</td>
<td></td>
<td>Comm.: 1</td>
<td>Enc.: 1</td>
<td>Dec.: 2</td>
<td>OWF: 2</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>Stateful</td>
<td>SSS + AE</td>
<td>Far (Fair)</td>
<td>Fair</td>
<td>8</td>
<td>$O(k + \lambda)$ bits</td>
<td></td>
<td></td>
<td>Comm.: 0</td>
<td>Enc.: 1</td>
<td>Dec.: 2</td>
<td>OWF: 0</td>
</tr>
<tr>
<td>KMG</td>
<td>Stateless</td>
<td>SSS + AE + MAC + ZKPoPK</td>
<td>Fair</td>
<td>Attack</td>
<td>2</td>
<td>$O(k + \lambda)$ bits</td>
<td></td>
<td></td>
<td>Comm.: 2</td>
<td>Encr.: 2</td>
<td>Dec.: 3</td>
<td>OWF: 2</td>
</tr>
<tr>
<td>$\Upsilon$</td>
<td>Stateless</td>
<td>SSS + AE</td>
<td>Far (Fair)</td>
<td>Far (Fair)</td>
<td>2</td>
<td>$O(k + \lambda)$ bits</td>
<td></td>
<td></td>
<td>Comm.: 1</td>
<td>Enc.: 2</td>
<td>Dec.: 3</td>
<td>OWF: 0</td>
</tr>
</tbody>
</table>
Thank you.