Kaires: Fully Decentralized Privacy-Preserving Machine Learning Framework

Islam Faisal* 1
@islamfaisalm

Eleftherios Kokoris-Kogias 2
@LefKok

Bryan Ford 2
@brynosaurus

Sherif El-Kassas 1
@kassas0

1 The American University in Cairo (AUC)
2 Ecole polytechnique fédérale de Lausanne (EPFL)
* Work mostly done while visiting EPFL.
Data Analytics in the Wild
Privacy Vs. Functionality
Access-Control
Robustness against “bad” clients
Robustness against “bad” clients
Robustness against “bad” servers

70
100
-42
System Goals

Access-Control
Users have revokable fine-grained access control over their data.

Privacy
Privacy of confidential data points as well as machine learning models.

Fairness
Fair exchange of data points and in-return value.

Robustness
Model can be built even with a failing (or dishonest) minority.

Decentralization
No single point of compromise or failure.

Auditability
Publicly verifiable tamper-proof access logs for data accesses.
TRUSTED THIRD PARTY

TRUSTED HARDWARE

APPLY CRYPTOGRAPHY
System Properties

- Confidentiality
- Auditability
- Atomic Data Delivery
- Dynamic Identity Management
- Decentralization
Machine Learning with CALYPSO

- Use CALYPSO to store and retrieve the data points
- Central machine learning node
- Data points are collected in a publicly auditable access-controlled system
- Data consumers have access to plain-text data points

<table>
<thead>
<tr>
<th>Property</th>
<th>CALYPSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access-Control</td>
<td>✓</td>
</tr>
<tr>
<td>Fairness</td>
<td>✓</td>
</tr>
<tr>
<td>Auditability</td>
<td>✓</td>
</tr>
<tr>
<td>Decentralization</td>
<td>✗¹</td>
</tr>
<tr>
<td>Privacy</td>
<td>✗</td>
</tr>
<tr>
<td>Robustness</td>
<td>✗</td>
</tr>
</tbody>
</table>

¹Access-control and secret-sharing are decentralized, but learning is centralized.
• A decentralized system for computing aggregate statistics
• Provides client-privacy as long as there is at least one honest server
• Servers learn about the data no more than they can learn from statistics
<table>
<thead>
<tr>
<th>Secret-Shared Non-Interactive Proofs (SNIPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Distributed zero-knowledge proofs that can prove whether a certain point ( x ) satisfies a boolean circuit ( \text{Valid}(x) )</td>
</tr>
<tr>
<td>• Provides robustness against adverserial clients</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-Party Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Local aggregators compute local values from shares</td>
</tr>
<tr>
<td>• A global aggregator combines all the local aggregators to obtain the model</td>
</tr>
</tbody>
</table>
System Design

- Combine CALYPSO with Prio to get a decentralized design
- Neither data consumers nor aggregation servers see the data in plain-text
- Extend Prio so that only the data consumer has access to the model
System Design

Data Provider

Data Consumer

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority
System Design

Data Provider

Publish Secret-Shared Data Point

Data Consumer

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority
System Design

Data Provider

Request Access for many points

Data Consumer

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority

Publish Secret-Shared Data Point

Request secret shares using the proofs

Grant access to authorized points

Local Aggregators

Combine all local aggregators
System Design

Data Provider

Grant access to authorized points

Data Consumer

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority
System Design

Data Provider

Data Consumer

Request secret shares using the proofs

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority
System Design

Data Provider

Data Consumer

Encrypted secret shares

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority
System Design

Data Provider

Data Consumer

Initiate Prio Protocol

Access-Control Cothority

Secret-Management Cothority

Aggregation Cothority
System Design

Data Provider

Data Consumer

Local Aggregators

Access-Control Cotherity

Secret-Management Cotherity

Aggregation Cotherity
System Design

- Data Provider
- Data Consumer

Combine all local aggregators

- Access-Control Cothority
- Secret-Management Cothority
- Aggregation Cothority
## System Design

<table>
<thead>
<tr>
<th>Property</th>
<th>CALYPSO</th>
<th>+Prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access-Control</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fairness</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Auditability</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Decentralization</td>
<td>?</td>
<td>✓</td>
</tr>
<tr>
<td>Privacy</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Robustness</td>
<td>✗</td>
<td>✓²</td>
</tr>
</tbody>
</table>

1. Access-control and secret-sharing are decentralized, but learning is centralized.
2. Robustness against adversarial clients only.
How can we make the system tolerate a faulty minority?
• SNIPs are essentially a multi-party computation of a certain arithmetic circuit $C_f$
• If only we could replace the circuit evaluation protocol by one that is fault-tolerant ...
Multi-party computation against a faulty minority

Creates a multi-party evaluation protocol based on a verifiable secret-sharing scheme (VSS)

**Verifiable Secret-Sharing (VSS)**

- **VSS Share**: Allows a dealer to share a certain with n nodes.
- **VSS Reconstruct**: Reconstruction protocol to be run by the nodes.

---

Multi-party computation against a faulty minority

Creates a multi-party evaluation protocol based on a verifiable secret-sharing scheme (VSS)

<table>
<thead>
<tr>
<th>Verifiable Secret-Sharing (VSS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• VSS Share: Allows a dealer to share a certain with ( n ) nodes.</td>
</tr>
<tr>
<td>• VSS Reconstruct: Reconstruction protocol to be run by the nodes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Secrets in VSS can be reconstructed with up to ( n/2 ) failing or dishonest nodes</td>
</tr>
<tr>
<td>• Computations in the MPC protocol based on VSS is robust against up to ( n/2 ) failing or dishonest nodes</td>
</tr>
</tbody>
</table>

### Properties Recap

<table>
<thead>
<tr>
<th>Property</th>
<th>CALYPSO</th>
<th>+Prio</th>
<th>+BFT Prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access-Control</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Fairness</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Auditability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Decentralization</td>
<td>?³</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Privacy</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Robustness</td>
<td>✗</td>
<td>✗⁴</td>
<td>✓</td>
</tr>
</tbody>
</table>

³Access-control and secret-sharing are decentralized, but the learning is centralized.
⁴Robustness against adversarial clients only.
Conclusion

- Designed a fully-decentralized fault-tolerant machine learning framework that is private, fair, auditable, and robust.
- The first system to our knowledge that achieves these properties without reliance on extra assumptions such as trusted hardware.
- Integrated the ByzCoin distributed ledger and Calypso service with the Prio MPC primitives to implement the system in Go.
Feedback is welcome!

Islam Faisal
Islam@bu.edu
@islamfaisalm

Eleftherios Kokoris-Kogias
@LefKok

Bryan Ford
@brynosaurus

Sherif El-Kassas
@kassas0