Collaborative attacks extend

Zizheng Liu, Bharat Bhargava

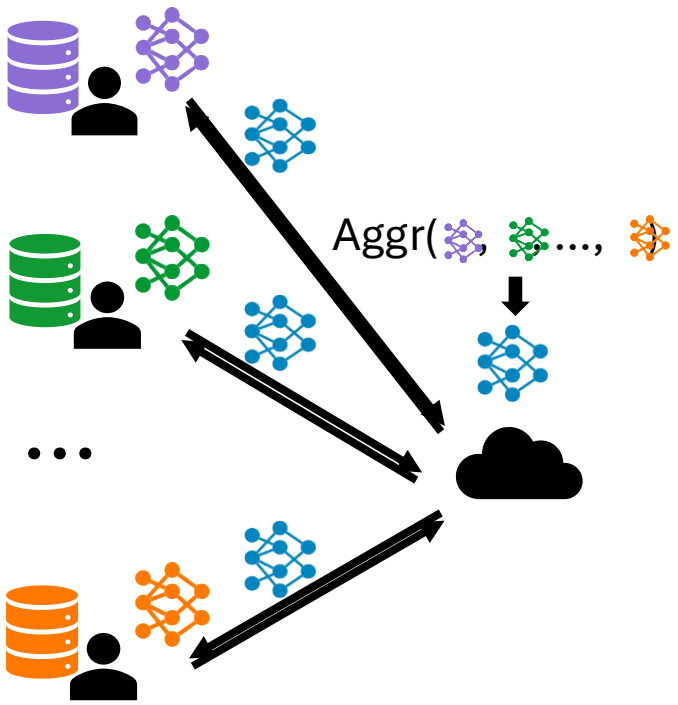
Robust Secure Aggregation for DFL

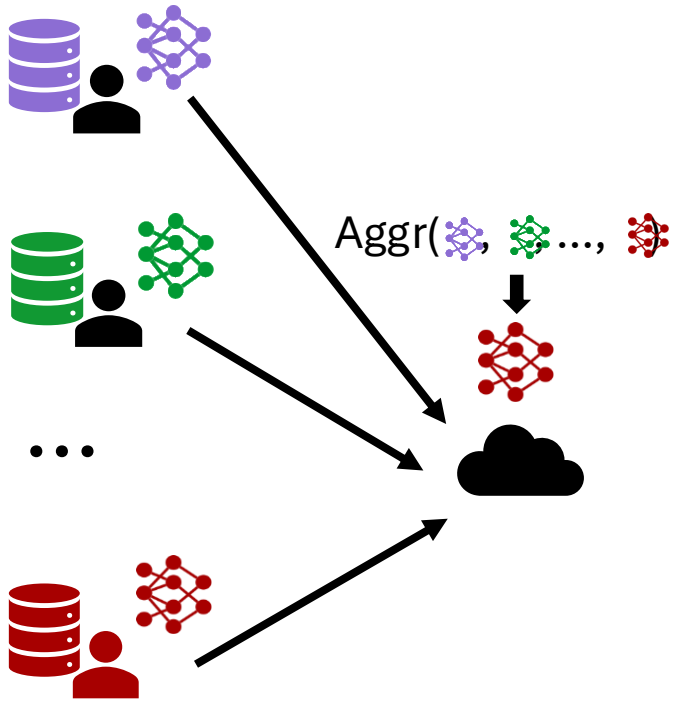
FL Security and Privacy

Federated Learning (FL)

Security: Poisoning attack

Privacy leakage



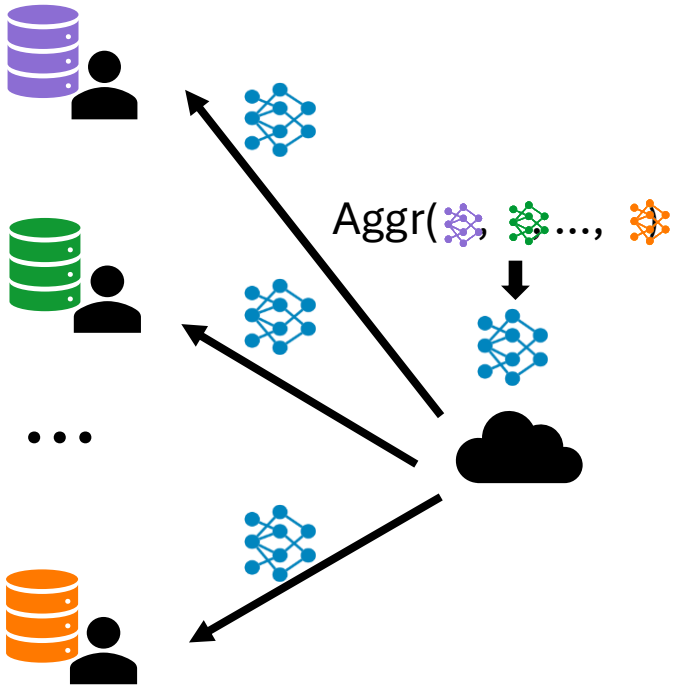


PURDUE UNIVERSITY

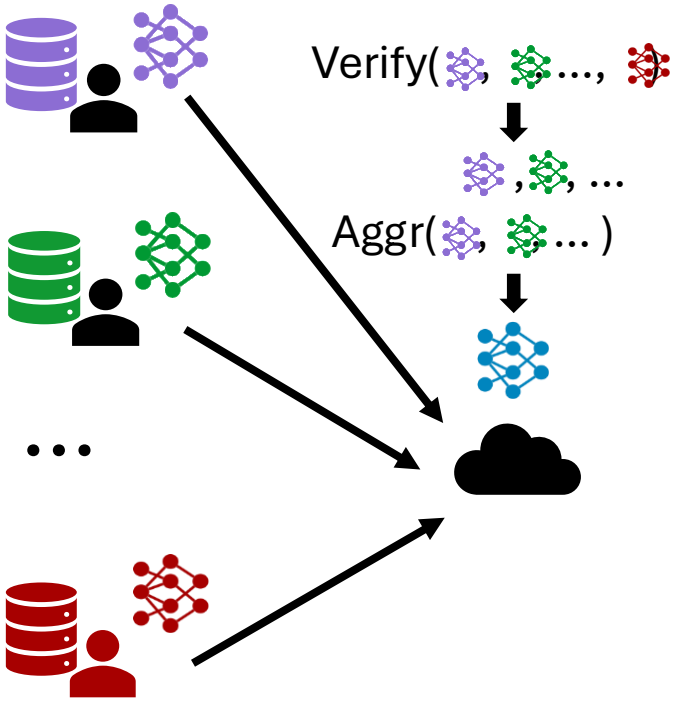
Robust Secure Aggregation for DFL

FL Security and Privacy

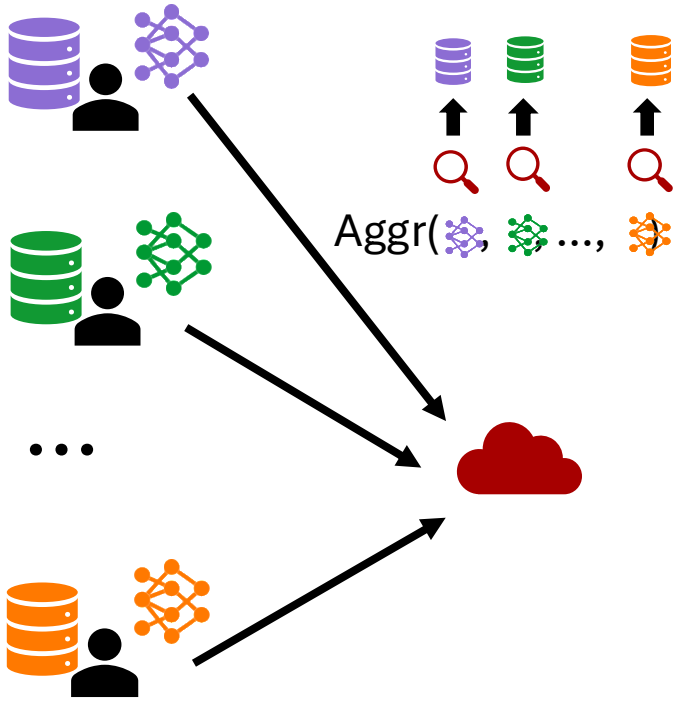
Federated Learning (FL)



Security: Poisoning attack



Privacy leakage



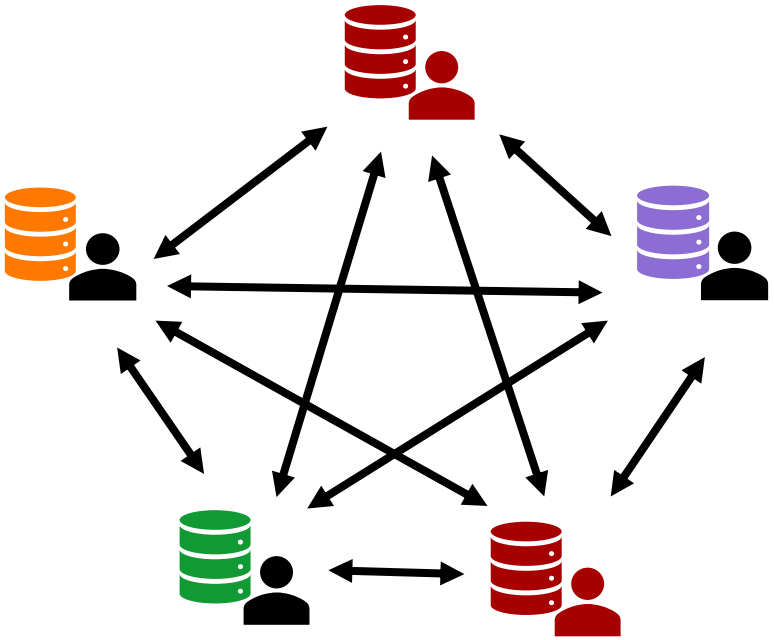
Verify(W_1, W_2,...,W_n) = W_1, W_2,...

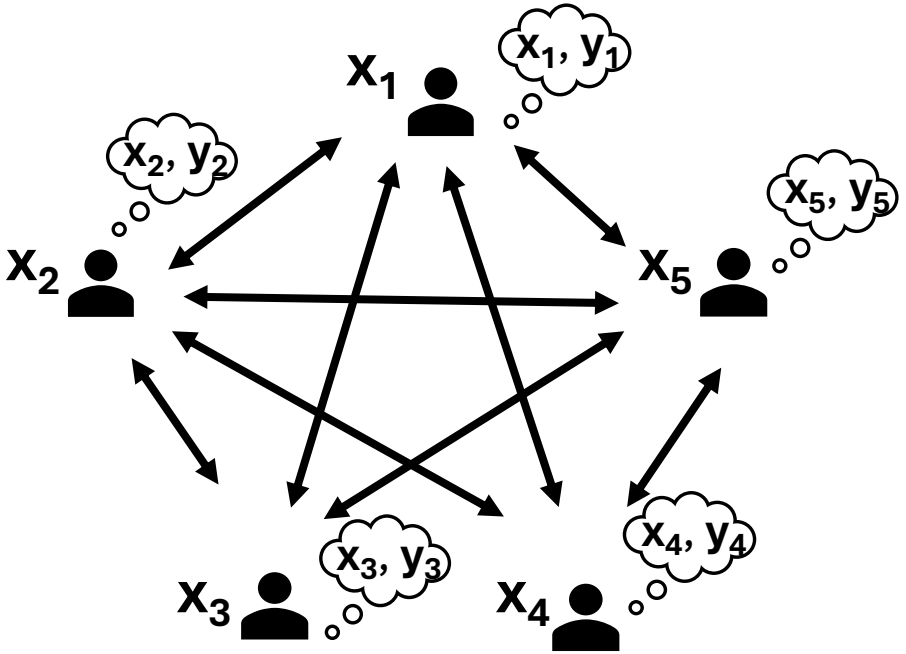
PURDUE UNIVERSITY

Robust Secure Aggregation for DFL

Decentralized FL and Secure Multi-party Computation

Decentralized Federated Learning MPC: Secure Multi-Party Computation





Verify(W_1, W_2,...,W_n) = W_1, W_2,...

Aggr(W_1, W_2,...,W_n) = G

$f(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}) = \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}, \mathbf{y_4}, \mathbf{y_5}$

Secure Aggregation: $f()$:= Aggr(); $\mathbf{x_i}$:= W_i; $\mathbf{y_1}$ = ... = $\mathbf{y_n}$:= G

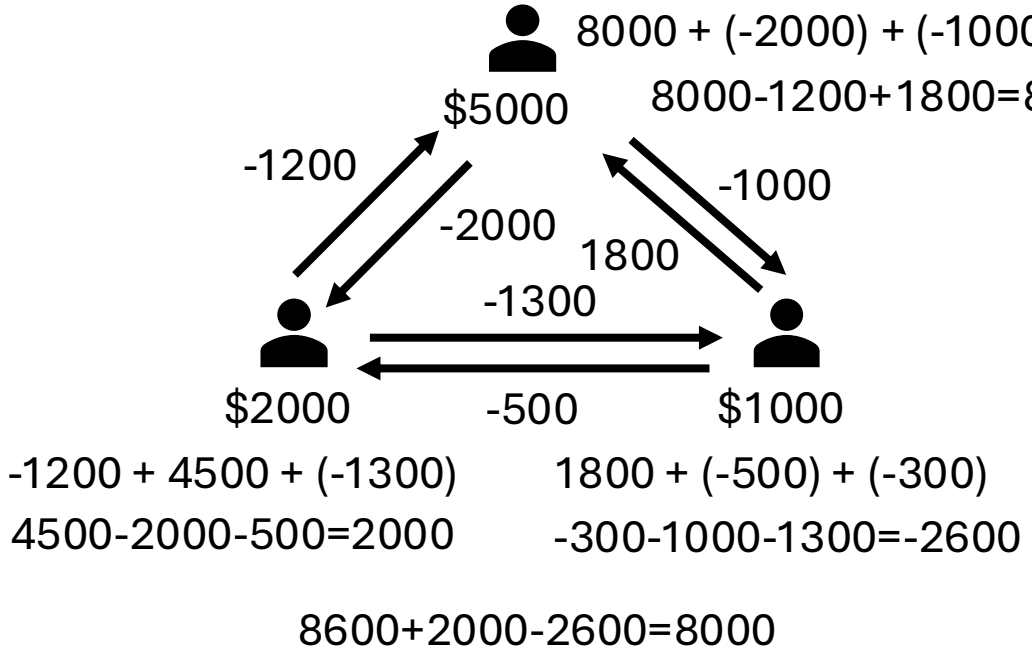
Robust Secure Aggregation: $f()$:= Aggr(Verify()); $\mathbf{x_i}$:= W_i; $\mathbf{y_1}$ = ... = $\mathbf{y_n}$:= G

PURDUE UNIVERSITY

Robust and Secure Aggregation for DFL

An example of MPC

MPC Example: compute average salary



Additive secret sharing: needs all secret shares to reconstruct the secret

- **If all participants are honest**, then by replacing salaries with model parameters and setting the aggregation function Aggr() to Avg(), the problem is resolved.
- If some participants submit malicious inputs, a Verify() function must be introduced.
  - Verify() involves more than simple addition; it requires additional computational operations.
- Beyond injecting malicious inputs, an adversary may also cheat during the secure computation process.
  - For example, the adversary may perform incorrect local computations to prevent the protocol from producing an output.
  - Alternatively, the adversary may simply refuse to participate in the computation.

PURDUE UNIVERSITY

Robust and Secure Aggregation for DFL

Threat Model

From Machine Learning Perspective

Security: Use **Verify()** to filter out malicious models.
Privacy: Use secure multiparty computation to aggregate models (execute **Aggr()**).

Security and Privacy:

- Use secure multiparty computation to execute **Verify()**, then perform **Aggr()** on the accepted models.

From MPC Perspective

Execute more operations

- Verify(W_t, W_r) = $\mathbb{I}\|W_t - W_r\|_2 \leq \rho$
  - $\|.\|_2$ needs multiplication and square root computation

Threat Models

- Semi-honest: all parties behave honestly to follow the protocol
- Malicious: parties deviate
  - Secure with abort: halt the protocol when cheating detected
  - Guaranteed output delivery: find the deviating party, exclude it and continue

Robust and Secure Aggregation for DFL

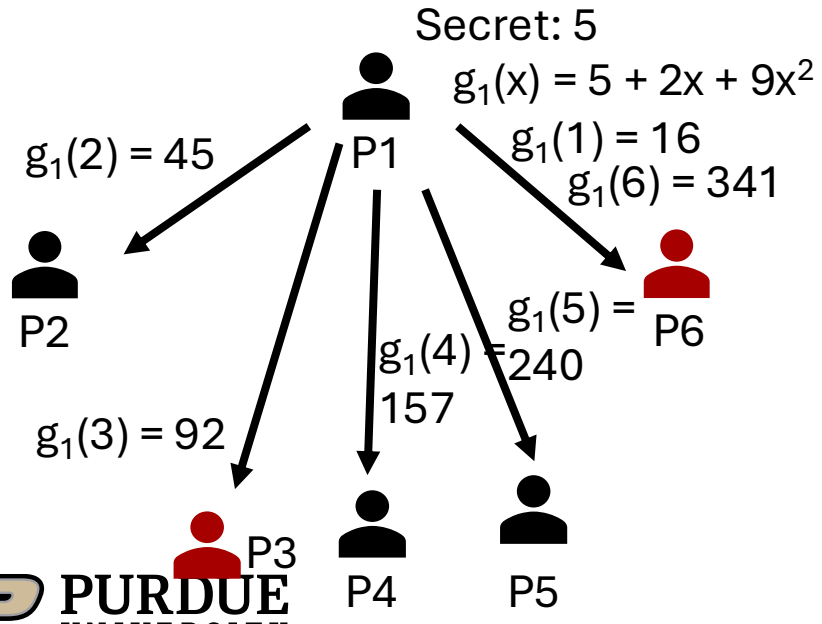
Shamir's Secret Sharing and DN multiplication Protocol

Additive secret sharing: needs all secret shares to reconstruct the secret

Shamir's n-t secret sharing

Any t+1 parties can reconstruct the secret

n = 6 parties, t = 2 are corrupted, honest majority, $2t + 1 \leq n$



Shamie SS Properties

- Denote $[x]_t$ as a degree-t sharing of x. [(1,16), …, (6,341)] is a degree-2 sharing of 5.
  - $[x]_t + [y]_t = [x+y]_t$
  - $k * [x]_t = [kx]_t$
  - $[x]_t * [y]_t = [x*y]_{2t}$ – needs 2t+1 shares to reconstruct the secret

DN Multiplication Protocol

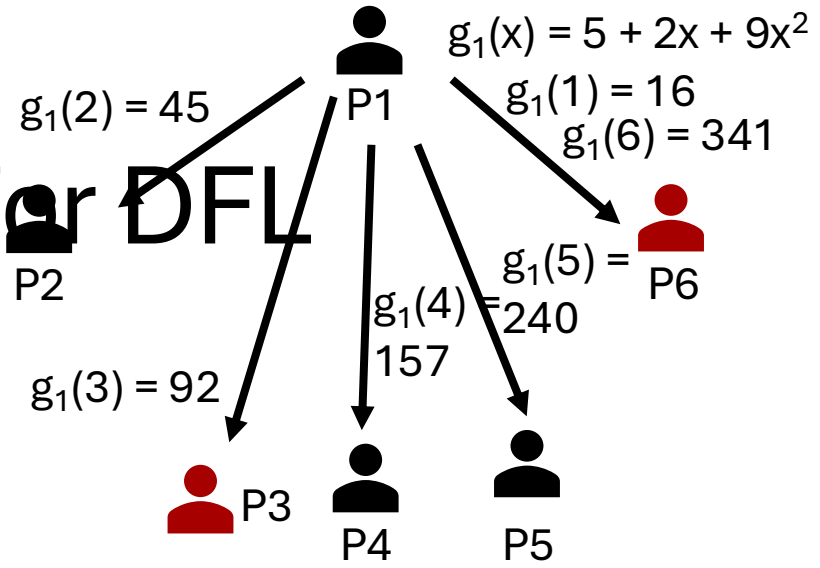
1. The parties prepare ([r]*t, [r]*2t), where r is a secret random value.
2. Each party locally computes [x]*t* * *[y]t = [xy]2t and sets [e]*2t := [xy]*2t* + *[r]*2t.
3. The parties select a coordinator Pking and send [e]2t to him.
4. Pking reconstructs e, generates [e]t, and distributes it to the parties.
5. Each party then locally computes [z]t = [xy]t = [e]t - [r]t.

During this process, any party, including Pking, may cheat.
This protocol provides only semi-honest security.

PURDUE UNIVERSITY

Robust and Secure Aggregation for DFL

Secure with Abort



Secure with Abort

The tuple ([x]_t, [y]*t, [r]t, [r]{2t}, [e]*{2t}, [e]_t, [z]_t) is called the **transcript** of the DN protocol. All parties must verify that every value in the transcript is consistent, i.e., that all corresponding shares are well-formed.

- Recall the example of Shamir secret sharing: the parties must ensure that all (id, share) pairs lie on the same polynomial, e.g., $5 + 2x + 9x^2$.
- However, directly revealing shares would compromise privacy.

Suppose $\{([x^{(i)}]_t, [y^{(i)}]_t, [r^{(i)}]_t, [r^{(i)}]_{2t}, [e^{(i)}]_{2t}, [e^{(i)}]_t, [z^{(i)}]_t)\}_{i=1}^{m}$ are the m transcripts that need to be verified.

1. The parties first run the DN protocol on random inputs $([x^{(0)}]_t, [y^{(0)}]_t)$ to obtain an additional transcript $([x^{(0)}]_t, [y^{(0)}]_t, [r^{(0)}]_t, [r^{(0)}]_{2t}, [e^{(0)}]_{2t}, [e^{(0)}]_t, [z^{(0)}]_t)$
2. Next, the parties generate a public random value λ. Each party locally computes
$$([\alpha]_t,[\beta]_t,[\delta]_t,[\delta]_{2t},[\eta]_{2t},[\eta]_t,[\gamma]_t) = \sum_{i=0}^{m} \lambda^i \, ([x^{(i)}]_t, [y^{(i)}]_t, [r^{(i)}]_t, [r^{(i)}]_{2t}, [e^{(i)}]_{2t}, [e^{(i)}]_t, [z^{(i)}]_t)$$
3. Each party then reveals its shares of $([\alpha]_t,[\beta]_t,[\delta]_t,[\delta]_{2t},[\eta]_{2t},[\eta]_t,[\gamma]_t)$, and the parties check whether all revealed shares lie on the same degree-t or degree-2t polynomial, respectively.

PURDUE UNIVERSITY

Robust and Secure Aggregation for DFL

Guaranteed Output Delivery

Guaranteed Output Delivery

1. For each share that a sender Pd distributes to a receiver Pi, denote it by $\sigma i(d)$. All other parties Pv act as **verifiers** for this share.
2. Each verifier Pv prepares an authentication key $\mu_{v \to i}$, for the receiver Pi. The verifier and the sender Pd then run a protocol to compute an authentication tag $\tau_{d,i,v} = \mu_{v \to i} * \sigma_i(d)$, and the result is delivered **only to Pi** (neither Pd nor Pv receives the output. That is, $x_d = \sigma_i(d)$, $x_v = \mu_{v \to i}$, $y_i = \tau_{d,i,v}$, and all other inputs and outputs are empty.
3. Whenever $\sigma_i(d)$ is disputed — for example, when all other shares lie on a certain polynomial but this one does not — Pi reveals the corresponding authentication tags to all verifiers. The verifiers then perform a **majority vote** to determine whether the sender Pd or the receiver Pi is cheating.
4. The cheating party is eliminated, and the computation is restarted.

PURDUE UNIVERSITY

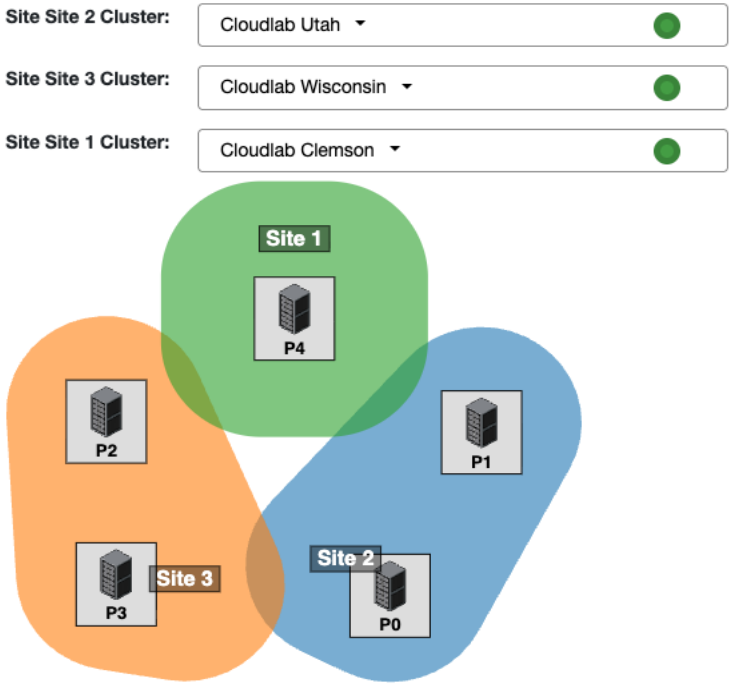
Robust and Secure Aggregation for DFL

Remainder questions and Experimental Results

- During the computation of authentication keys and authentication tags, parties may still behave maliciously.
- When a party is eliminated due to detected misbehavior, the number of remaining parties may fall below 2t+1, in which case degree-2t secret sharings can no longer be reconstructed.
- In addition to addition and multiplication, protocols are required for other types of operations as well.
- Machine learning computations typically use floating-point numbers, whereas secret sharing schemes operate over fixed-point representations.
- All of the above issues are addressed by specialized secure multiparty computation protocols. This project involves approximately 40 such protocols.

Implementation: Based on the semi-honest DN multiplication protocol in MP-SPDZ, we incorporated additional operation logic and ensured that all operations provide guaranteed output delivery under malicious behavior. The implementation consists of 3,000+ lines of C++ code.

The protocol was successfully executed in a distributed network with nodes located in Utah, Wisconsin, and South Carolina.



PURDUE UNIVERSITY