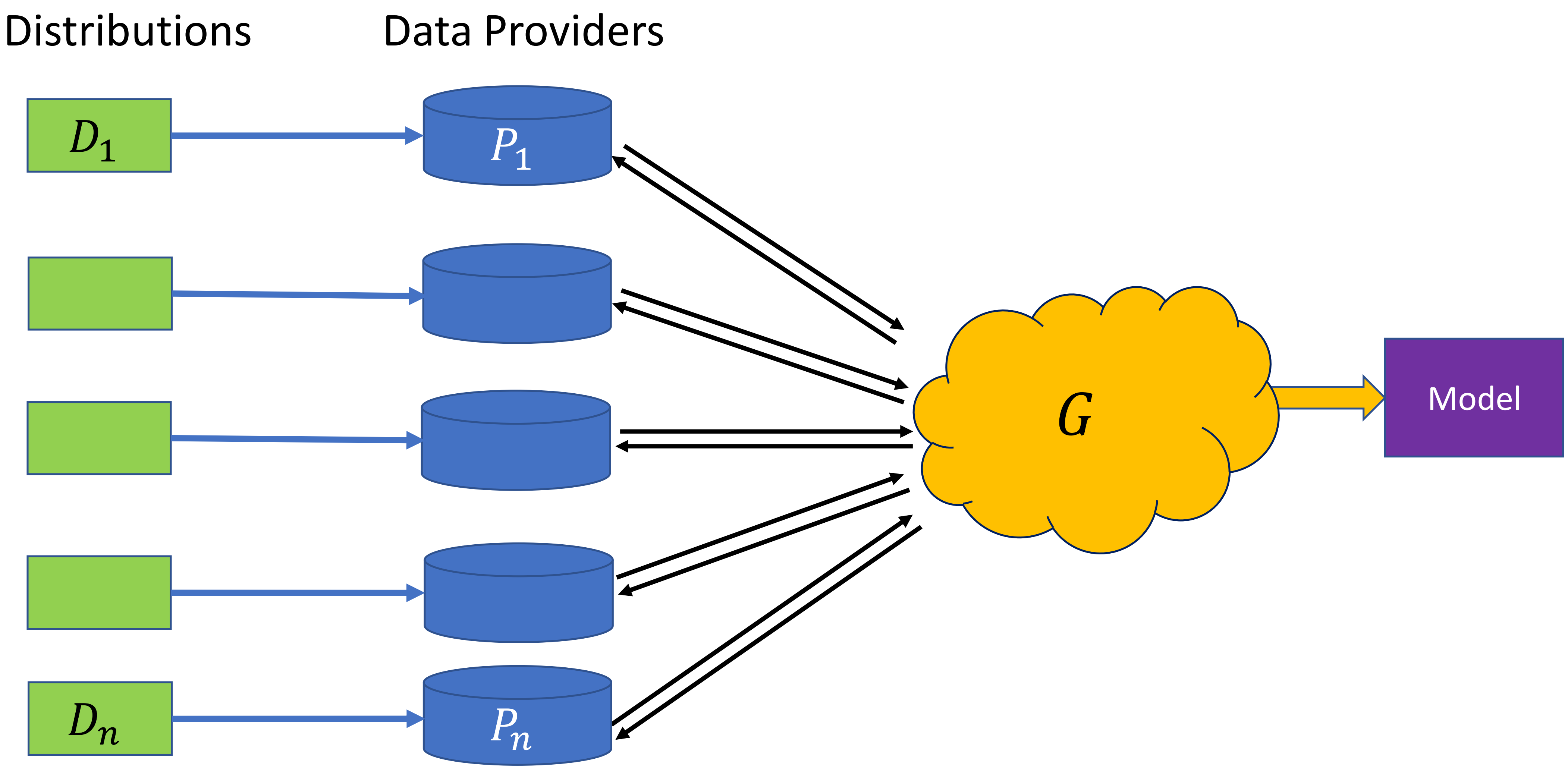
Universal Multi-Party Poisoning Attacks

Saeed Mahloujifar

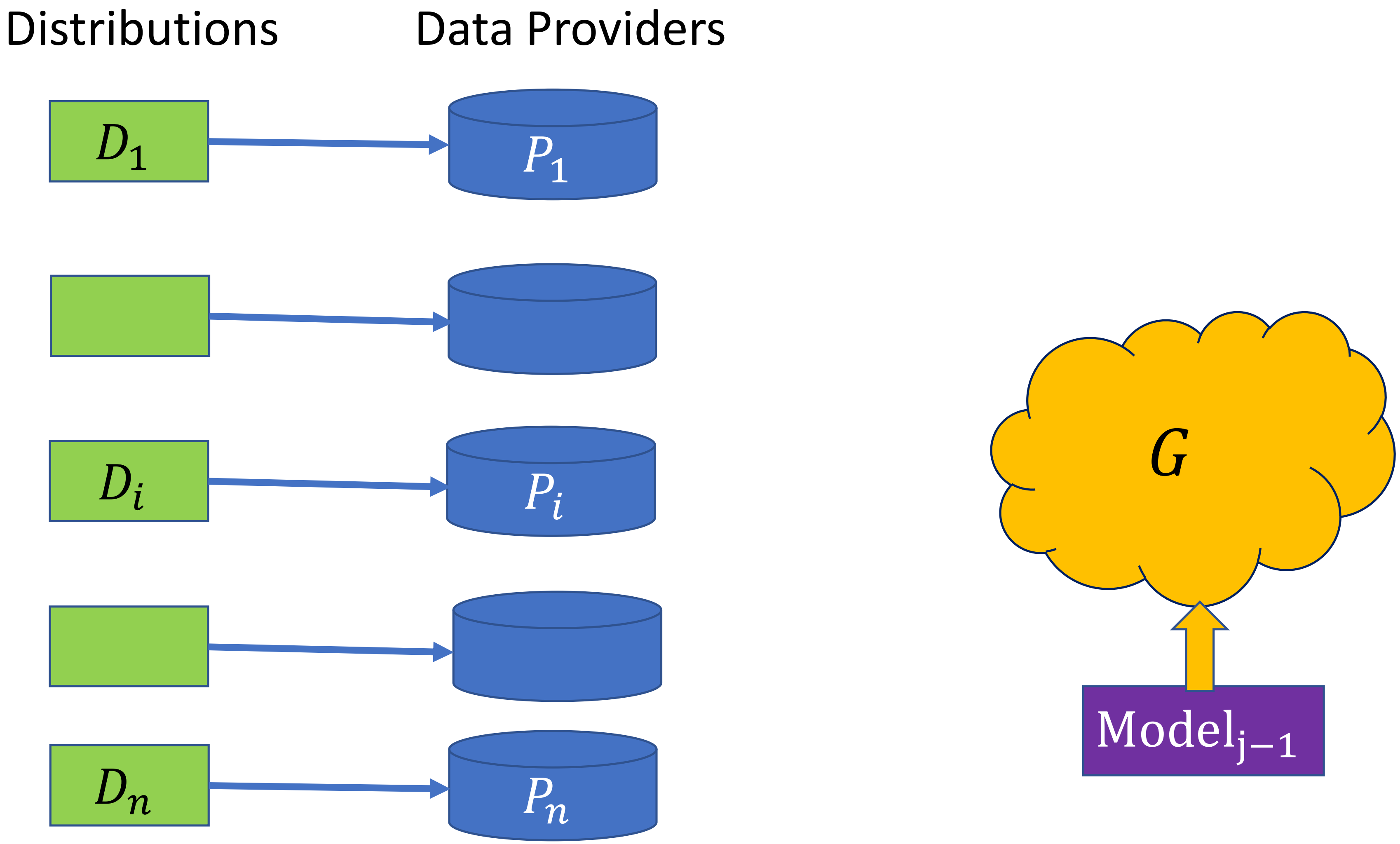
Mohammad Mahmoody

Ameer Mohammed

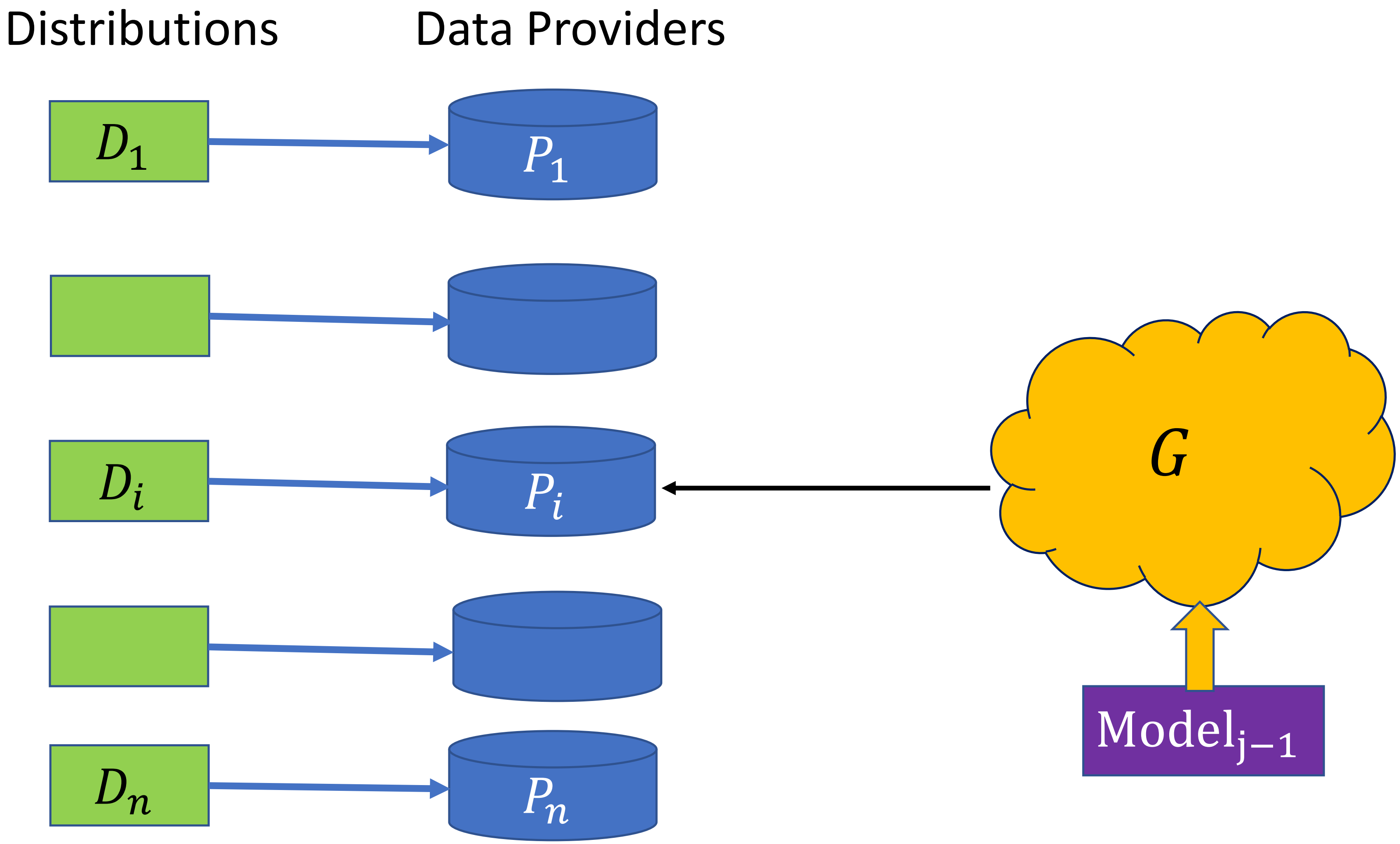
Multi-Party Learning



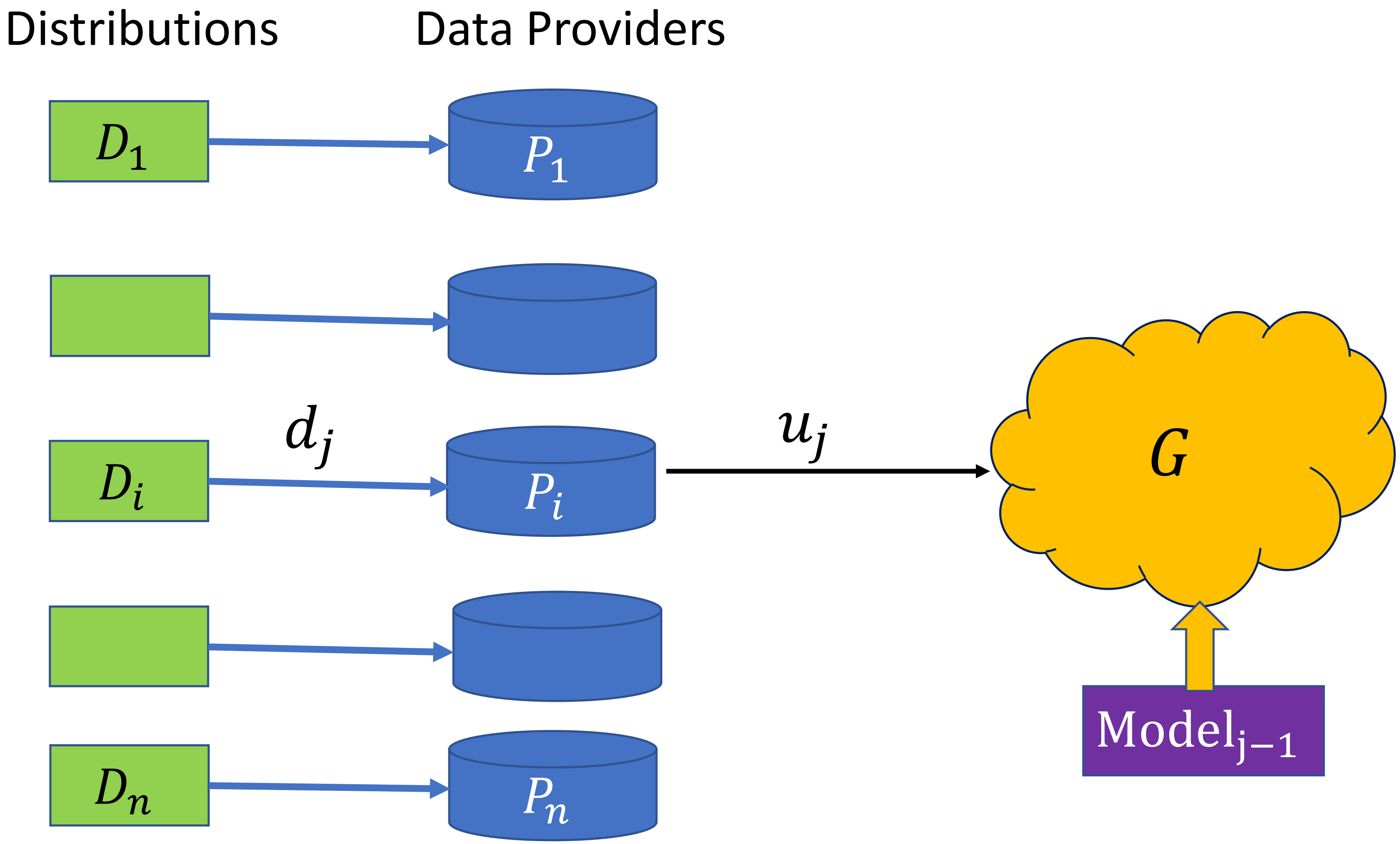
Multi-Party Learning (Round j)



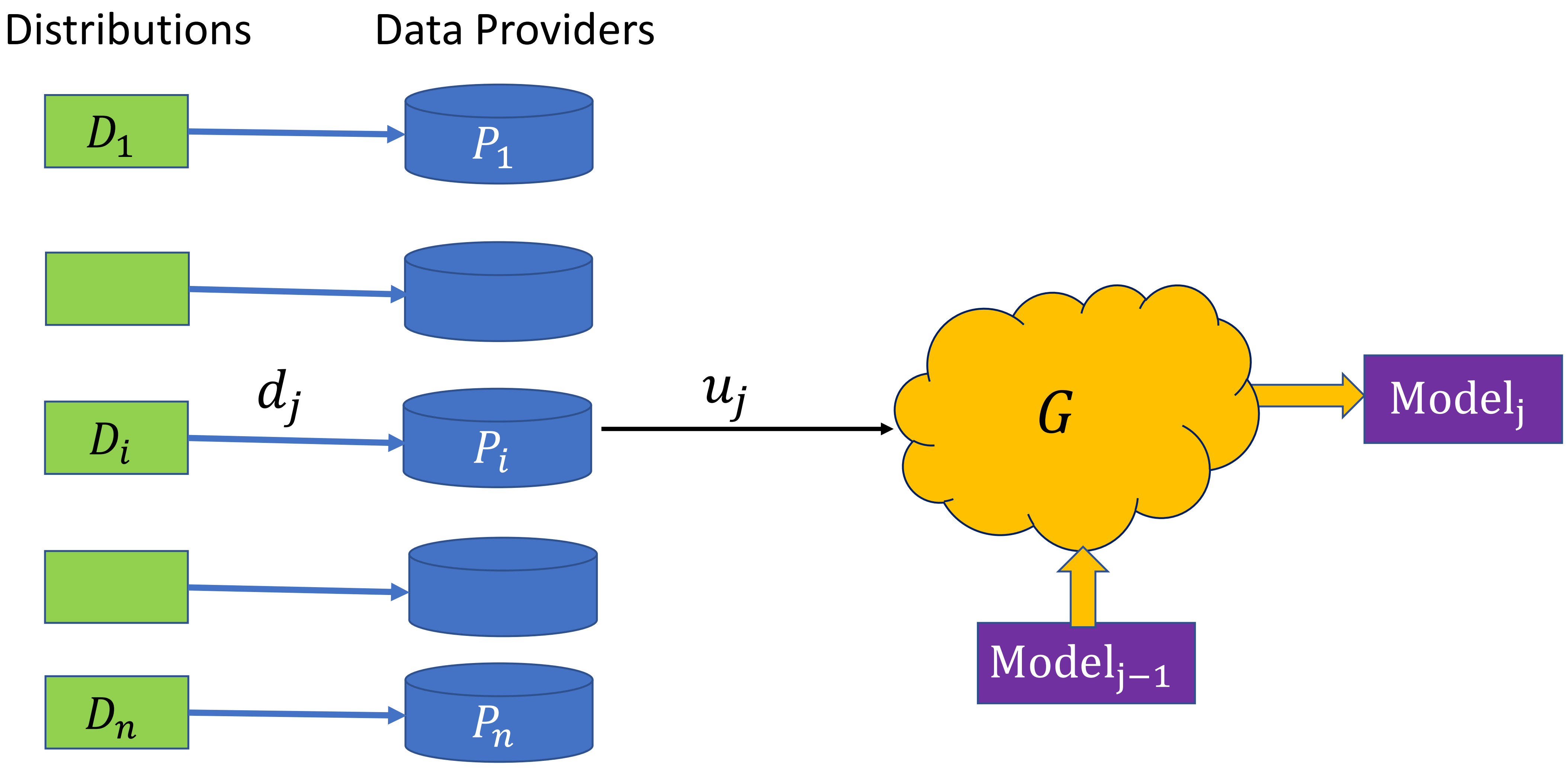
Multi-Party Learning (Round j)



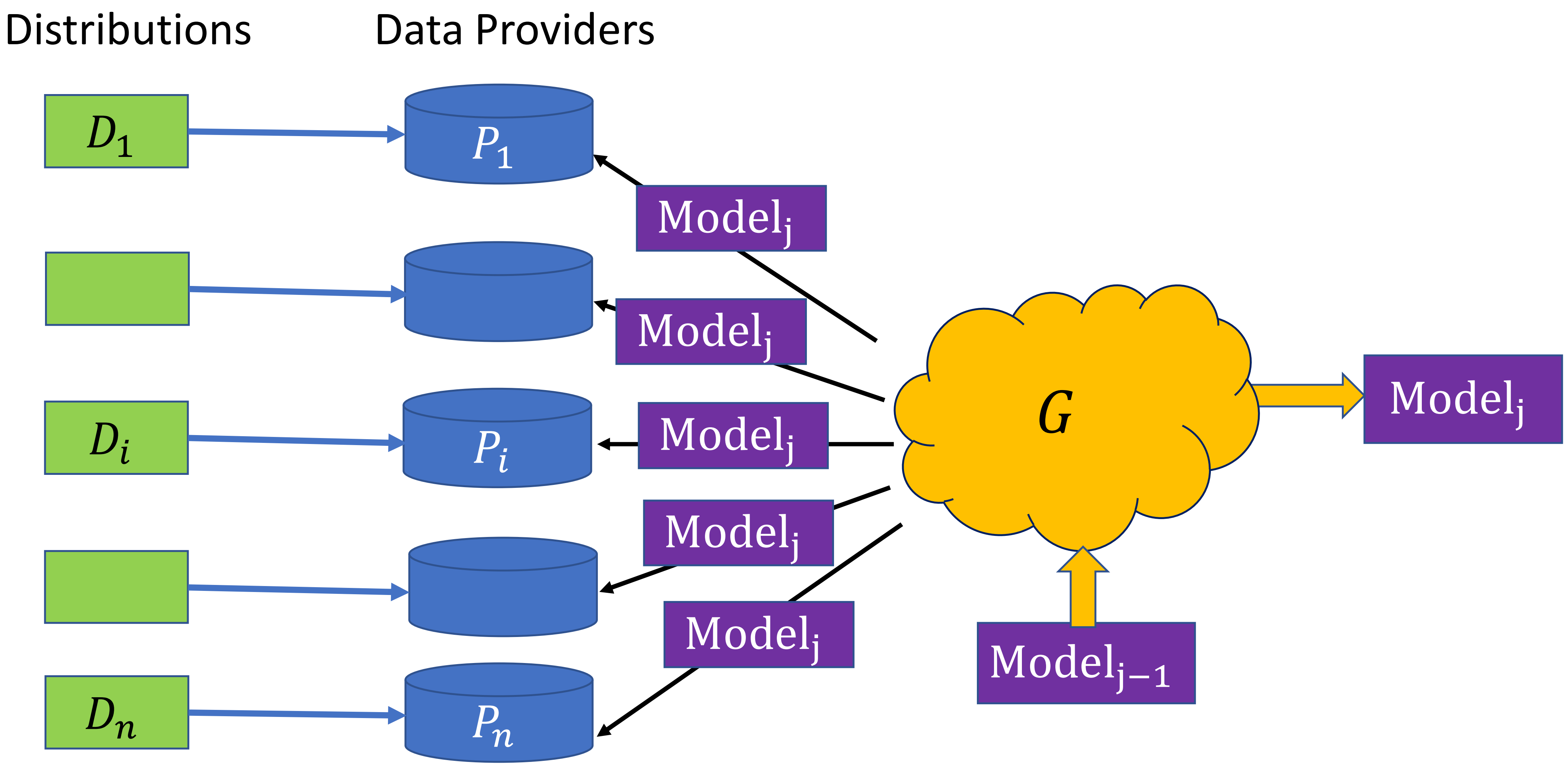
Multi-Party Learning (Round j)



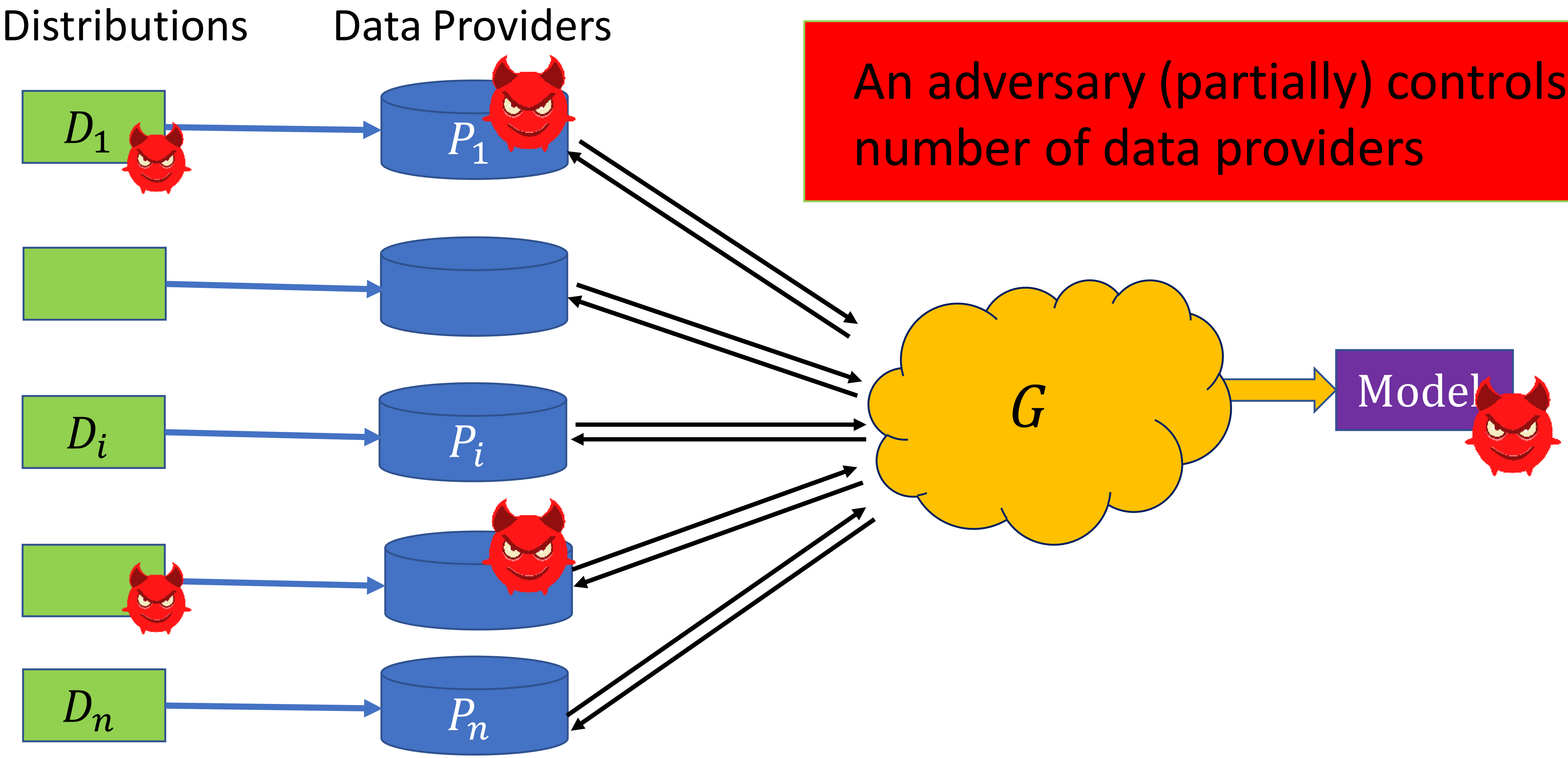
Multi-Party Learning (Round j)



Multi-Party Learning (Round j)



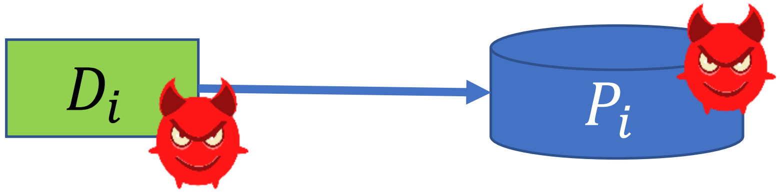
Poisoning in Multi-Party Learning



An adversary (partially) controls a number of data providers

(k, q)-Poisoning Attack Model

k (out of n) of the parties become corrupted



Each corrupted party P_i samples from a different distribution

$$d(\boxed{D_i}, \boxed{D_i}) \leq q$$

$k = n \;\rightarrow\; q$-Tampering [ACMPS14] [MM17] [MM18]
$q = 1 \;\rightarrow\;$ Static Corruption in MPC (crypto)

What is the inherent power of (k, q)-poisoning adversaries against Multi-party Learning?

Main Theorem: Power of (k, q)-Poisoning

Let B be a bad property of the model M

- E.g. $B(M) = 1$ if M misclassified an specific instance x

For any n-party learning protocol there is a (k, q)-poisoning adversary that increases $\Pr[B]$ from

$$\epsilon \to \epsilon^{1-\frac{kq}{n}}$$

Main Theorem: Power of (k, q)-Poisoning

Let B be a bad property of the model M

- E.g. $B(M) = 1$ if M misclassified an specific instance x

For any n-party learning protocol there is a (k, q)-poisoning adversary that increases $\Pr[B]$ from

$$\epsilon \to \epsilon^{1-\frac{kq}{n}}$$

$\Pr[B]$ Before attack	q	k	$\Pr[B]$ after attack
5%	$1/2$	$n/2$	11%
5%	$1/2$	n	22%
5%	1	$n/2$	22%

Features of Attack

- **Universal:** provably work against any learning protocol
  - In contrast with: [Bagdasaryan et al 2018; Bhagoji et al. 2018]
- **Clean label:** Only uses correct labels
  - Similar to: [M et al 2017; Shafahi et al 2018]
- **Polynomial time**
  - Similar to: [M and Mahmoody 2019]

Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing

Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing
- New biasing model: Generalized p-Tampering.

Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing
- New biasing model: Generalized p-Tampering.

Let $f : (U_1, \dots, U_n) \to \{0,1\}$

Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing
- New biasing model: Generalized p-Tampering.

Let $f : (U_1, \ldots, U_n) \to \{0,1\}$

Input blocks $u_1, u_2, \ldots u_n$ are sampled one-by one in online way:

Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing
- New biasing model: Generalized p-Tampering.

Let $f : (U_1, \ldots, U_n) \to \{0,1\}$

Input blocks $u_1, u_2, \ldots u_n$ are sampled one-by one in online way:

$u_i =$ U_i with marginal probability $1 - p$

 with marginal probability p

Ideas Behind Attack

- Main Idea: Treat protocol as random process and run a biasing attack
  - The bad property is a function over the random process
  - We want to bias that function, similar to attacks in coin tossing
- New biasing model: Generalized p-Tampering.

Let $f : (U_1, \ldots, U_n) \to \{0,1\}$

Input blocks $u_1, u_2, \ldots u_n$ are sampled one-by one in online way:

$u_i =$

- U_i with marginal probability $1-p$
-  with marginal probability p

Our generalized p-tampering attack based on Ideas in coin tossing attacks [BOL89,IH14]

Summary

We show Poisoning attacks against multi-party learning protocols:

- **Universal:** Provably apply to any multi-party learning protocol
- **Clean label:** Only uses samples with correct labels
- Run in **polynomial time**
- Increase the probability of **any chosen bad property**

Poster #160