Imagine a Kaggle competition with $n$ competitors and have a winner declared, but would like to keep their proprietary algorithms and output secret. How can we provide this cryptographic functionality?

### A Friendly Competition

There are a number of working commitment schemes, each with unique and the public connector as a common scoring algorithm. The competitors would like to participate and have a winner known as n-party private, or PACCMulE: Private Aggregate Circuit Commitments for Multiple Entities

To formalize the problem, we introduce a class of arithmetic circuits

Formalization: N-Party Private Circuits

- **A Friendly Competition**
- **Formalization: N-Party Private Circuits**
- **Proof of Functional Relation**
- **PACCMulE Objectives**
- **Desired Security Properties**
- **Implications on State of the Art**
- **Future Research Directions**

### PACCMulE: Private Aggregate Circuit Commitments for Multiple Entities

We are trying to create an analogous commitment scheme in this multi-party setting. Formally, we define a PACCMulE for a $n$-party private circuit $A = (\cup_{i=1}^{n} p_i)A_i$, where $A$ is a tuple (Setup, Commit, Eval) where:

- **Setup**: $(\lambda \{X_1, \ldots, X_n\}) \rightarrow \lambda$: Sample public parameters.
- **Commit**: $(pp, A \in AC, c \in C)$: Produce a commitment of $A$.
- **Eval**: $PV(pp, A, r, x, \ldots, y)$: For $r \in X \times Y \times \mathbb{G}$, $(pp, A, r, x, y \rightarrow \{0, 1\})$. Convince all parties that $A(x) = y$.

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]

A circuit $A$ is then defined with the following polynomials, which are in-

\[ y \left( p_1 \cdots p_n \right) = \begin{cases} 1 & \text{if gate } i \text{ is on} \\ 0 & \text{otherwise} \end{cases} \]