Multiparty Computation:
Set of n parties jointly want to compute a function
→ preserving privacy of their inputs
→ some of the parties can be adversarial

Applications:
Elections
Auctions
Encrypted search
Privacy preserving data mining
Secure set intersection

Elections:
→ Should not change vote.
→ Privacy of my vote.
→ Correctness: I exactly (up to 1 vote/person)
    every vote casts belong to a unique voter.
→ Fairness:
Generic vs. tailored approaches:

MPC protocol for elections, encrypted search;
- use case by use case basis
- generic approach

-----

Types of adversaries:

- semi-honest: follow the protocol; (honest but curious)
- malicious: behave arbitrarily, computationally bounded
- covert: avoid to being caught
- Crashing: can stop participating in the protocol (fail-stop)

Corruption strategies:

- static: each party is assigned to be honest adversarial at the start of the protocol.
- adaptive: can corrupt a party during execution.

Execution setting

Standalone execution:
Concurrent execution: UC framework.

Communication model:

- Communication graph: all-to-all
- Authenticated & private
- Message delivery: synchronous, asynchronous
- Broadcast channel: all parties will receive the same message

Feasibility results: \( p > n/2 \)
- If we have an honest majority, any
function can be computed even against adaptive adversaries.

- W/O honest majority, it is impossible to achieve fairness.

- W/O honest majority, no fairness, it is possible to compute any functionality.

Expressing functions

Ideal world / Real world paradigm

\[ f(x_1, x_2, x_3) \]

\[ \uparrow \]

Ideal World: Input of p, Real world:
\[ S \rightarrow f(x_1, x_2, x_3) \]

ideal world adversary/Simulator

\[ \text{Ideal}_{f,S}(k, \overline{x}, z) \]

\[ \text{Real}_{\Pi,A}(k, \overline{x}, z) \]

\[ \forall \text{adversary } A; \exists S \text{ Distinguisher} \]

\[ R_k[\mathcal{D}(\cdot = 1)] - R_k[\mathcal{D}(\cdot = 1)] \leq \text{negl}(k). \]