Secure multiparty computation with malicious adversaries
Principle

- There is a (randomized) function $f : (\{0, 1\}^\ell)^n \rightarrow (\{0, 1\}^\ell)^n$.
- There are $n$ parties, $P_1, \ldots, P_n$.
  - Some of them may be adversarial.
- Party $P_i$ has the bit-string $x_i \in \{0, 1\}^\ell$.
- Party $P_i$ wants to learn $y_i$, where

  \[(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)\, .\]

- There is an access structure $\varnothing$, listing the intolerable coalitions
- Let $I \subseteq \{1, \ldots, n\}$. If $I \not\in \varnothing$, then the coalition $\{P_i\}_{i \in I}$ may not learn anything beyond $\{(x_i, y_i)\}_{i \in I}$
Real vs. Ideal security definitions

$(P_1, \ldots, P_n)$ is at least as secure as $I$

$$\exists Sim \forall Z \forall A : \text{view}_{\text{REAL}}(Z) \approx \text{view}_{\text{IDEAL}}(Z)$$
Details of machines $P_i$

- Goal: compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$
- At beginning, may receive “corrupt” from $A$
- May do initialization together with other $P_1, \ldots, P_n$
- Receives $x_i$ from $Z$
  - if corrupt, sends $x_i$ to $A$, receives back updated $x_i$
- Runs the protocol with other $P_1, \ldots, P_n$
  - Honest parties follow the instructions
  - Corrupted parties send all received messages to $A$, receive back messages to send to other parties
- Eventually obtains the result $y_i$
  - if corrupt, sends $y_i$ to $A$, receives back updated $y_i$
- Sends $y_i$ to $Z$
Ideal functionality for secure MPC

- At the beginning, receives “corrupt $C$” for $C \subseteq \{1, \ldots, n\}$ from the adversary
  - If $C \in \emptyset$, then put $C := \{1, \ldots, n\}$
- Receives $x_1, \ldots, x_n$ over respective connections
  - If $i \in C$: sends $x_i$ to adversary
  - Eventually, adversary may give back updated $x_i$
- If all values $x_1, \ldots, x_n$ are present, then compute
  $$(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$$
- Give $y_i$ to the adversary for all $i \in C$; the adversary gives back updated $y_i$
- Give $y_i$ to $\mathcal{Z}$ over respective connections
Variation: active-with-abort security

- At the beginning, receives "corrupt $C$" for $C \subseteq \{1, \ldots, n\}$ from the adversary
  - If $C \in \emptyset$, then put $C := \{1, \ldots, n\}$
- Receives $x_1, \ldots, x_n$ over respective connections
  - If $i \in C$: sends $x_i$ to adversary
  - Eventually, adversary may give back updated $x_i$
- If all values $x_1, \ldots, x_n$ are present, then compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$
- Give $y_i$ to the adversary for all $i \in C$; the adversary gives back updated $y_i$
- For each $i \in \{1, \ldots, n\}$: if the adversary allows, then give $y_i$ to $Z$ over respective connection
Variation: covert security

- At the beginning, receives “corrupt \( C \)” for \( C \subseteq \{1, \ldots, n\} \) from the adversary
  - If \( C \in \emptyset \), then put \( C := \{1, \ldots, n\} \)
- Receives \( x_1, \ldots, x_n \) over respective connections
  - If \( i \in C \): sends \( x_i \) to adversary
  - Eventually, adversary may give back updated \( x_i \)
- If all values \( x_1, \ldots, x_n \) are present, then compute
  \[(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)\]
- If it receives “cheat \( j \)” with \( j \in C \) from the adversary, then
  - With probability \( \epsilon \), cheating failed
  - With probability \( (1 - \epsilon) \), cheating succeeded
    Adversary gets notified of the outcome.
- next steps depend on the existence, success or failure of the cheating attempt
Covert security follow-up

- Give $y_i$ to the adversary for all $i \in C$; the adversary gives back updated $y_i$
- If cheating failed: Set $y_i := \text{“cheating } j\text{”}$ for all $i \notin C$
- If no successful cheating: Give $y_i$ to $\mathcal{Z}$ over respective connections
- If successful cheating: Give $y_i$ to $\mathcal{Z}$ only if the adversary allows it
  - The adversary may also change $y_i$
- If there was an attempt to cheat, then: Possibly: give to the adversary also $(x_i, y_i)$, where $i \notin C$
Broadcast

The computed function

\[ f(v, \bot, \ldots, \bot) = (v, v, \ldots, v) \]

The meaning

- One party (the sender) inputs a value \( v \)
- All honest parties output the same value \( v' \)
- If the sender was also honest, then \( v' = v \)

Studied settings

- Network — synchronous or asynchronous
- Cryptography — used or not used
We request an extension

\[\{\text{An extension would be bad}\}\]_K_B
(Im)possibility results

\( n \) — total number of parties; \( f \) — number of corrupt parties

- Asynchronous network — there is no deterministic protocol for broadcast
  - Randomized protocols exist for \( f < \frac{n}{3} \)
- Synchronous network:
  - No cryptography — \( f < \frac{n}{3} \) is necessary and sufficient
  - With cryptography — \( f \leq n - 2 \) is possible
A protocol for $f < n/3$ (1/2)

The protocol

- Use the protocol $\Pi_f(P, S, v)$, where
  - $P$ is the set of all parties, $|P| = n$
  - $S \in P$ is the sending party
  - $v$ is the value to be sent

Protocol $\Pi_0(P, S, v)$

- $S$ sends $v$ to all parties in $P \setminus \{S\}$
- $S$ outputs $v$. $P \in P \setminus \{S\}$ outputs the received value, or $\perp$, if none
A protocol for $f < n/3$ (2/2)

\[
\text{majority}(v_1, \ldots, v_k) := \begin{cases} 
  v, & \text{if } v_i = v \text{ for more than } k/2 \text{ values of } i \\
  \bot, & \text{otherwise}
\end{cases}
\]

**Protocol $\Pi_m(P, S, v)$**

- $S$ sends $v$ to all parties in $P \setminus \{S\}$
- For each $R \in P \setminus \{S\}$: invoke $\Pi_{m-1}(P \setminus \{S\}, R, v)$
- $S$ outputs $v$
- Each $R \in P \setminus \{S\}$ obtained $v_1, v_2, \ldots, v_{|P|-1}$ from the invocations of protocols $\Pi_{m-1}$. Outputs $\text{majority}(v_1, \ldots, v_{|P|-1})$
Security proof

- **Lemma.** If $|P| \geq 2k + m$, num. of bad parties is at most $k$, and $S$ is honest, then all honest parties output $v$ in $\Pi_m(P, S, v)$
  - Induction over $m$

- **Theorem.** If num. of bad parties is at most $m$, and $|P| \geq 3m + 1$, then $\Pi_m$ is a secure protocol for broadcast
  - Case “Honest sender” is covered by the lemma
  - Case “Bad sender” is proved by induction over $m$
A protocol with signatures

- Receiver $R_i$ initializes $V_i := \emptyset$
- Sender $S$ sends $\{v\}_S$ to all receivers
- Whenever some $R_i$ receives $M = \{\cdots \{\{v'\}_S\}_S\}_R_1 \cdots\}_R_k$:
  - Ignore it, if it has seen the sequence of signers $S, R_1, \ldots, R_k$ earlier
  - Ignore it, if $S, R_1, \ldots, R_k$ are not all different
  - If $v' \not\in V_i$, then
    - Update $V_i := V_i \cup \{v'\}$
    - If $k < f$, then send $\{M\}_{R_i}$ to everybody except $S, R_1, \ldots, R_k$

- At the end, $R_i$ outputs $\text{choice}(V_i)$
  - $\text{choice}$ is any deterministic function from sets of messages to messages (or $\bot$), satisfying $\text{choice}(\{v\}) = v$ for any message $v$
Security proof

Lemma. If $R_i$ and $R_j$ are honest, then $V_i = V_j$

- Sufficient to show: if $v' \in V_i$ then $v' \in V_j$
- If $v' \in V_i$, then $R_i$ received $[[\cdots [[v']_S]_{R_1} \cdots]_{R_k}$
  - If $k < f$, then $R_i$ forwarded it to $R_j$
  - If $k = f$, then one of $S, R_1, \ldots, R_k$ was honest. This party sent $v'$ to $R_j$, too

Lemma. If $S$ is honest, then $V_i = \{v\}$ for each honest $R_i$
Impossibility result (no crypto)

Theorem

There is no secure broadcast protocol for three parties that tolerates a malicious party

Corollary

There is no secure broadcast protocol for \(3f\) parties that tolerates \(f\) malicious parties

Proof of the corollary

- Suppose \(\Pi_f\) is such a protocol
- Construct a three-party protocol, by
  - The sender playing the sender and \((f - 1)\) receivers of \(\Pi_f\)
  - Each receiver playing \(f\) receivers of \(\Pi_f\)