UP TO NOW

- **Two-message two-party protocols**
  - Can compute everything in 2 messages, good comm.
  - Can be extremely slow

- **Multi-message two-party protocols**
  - Same, but faster
  - Needs many rounds
MULTI-PARTY COMPUTATION

- MULT/EQ/... protocols generalize to \( n \) parties
  - assuming appropriate secret sharing
- Both computational and information-theoretic setting possible
  - **Comp. setting:** secure if somebody is honest
  - **Inf.-th. setting:** secure if majority is honest
TODAY

1. Generalizing the last lecture to $n > 2$ parties
   - Security based on DDH (and trust: $t$ honest)
     Can take $t = 1$ (need to only trust yourself)

2. Information-theoretically secure protocol
   - Security based on trust alone (majority honest)
     Need: $t > n / 2$ honest
   - BGW protocol (Ben-Or, Goldwasser, Wigderson)

Note: since we are interested in semihonest model, here by honest we mean that we trust that no $t$ parties (no $t$-coalition) are going to pool their inputs together
(N,T)-THRESHOLD ENCRYPTION

- Assume we have \( n > t \) parties
- Every party generates her random key \( y_i \) and public key \( h_i \)
- Let \( h \) be the joint public key
- **Completeness:**
  - given access to \( h, \text{Enc}_h(m) \), and at least \( t \) different secret keys, one can efficiently recover \( m \)
- **IND-CPA under \((t - 1)\)-corruption:**
  - given access to \( h, \text{Enc}_h(m) \), and at most \( t - 1 \) different secret keys, an efficient adversary cannot break IND-CPA
IDEA: SECRET SHARING

- **(2,2) case:** $y = y_0 + y_1 \pmod{q}$
- **Completeness:** From $y_0$ and $y_1$ one can recover $y$
- **Hiding:** If $y_i$ is random, then just knowing $y_{1-i}$ does not give any information about $y$
- We need the same for arbitrary $(n, t), t < n$
QUIZ: (N, N)-SECRET SHARING

- Assume the case \((n, t = n)\)

- **Question:**
  - how to combine \(y\) from \((y_1, ..., y_n)\) so that no \(n - 1\) parties can obtain any information about \(y\)?
Assume the case \((n, t = n)\)

**Question:**

- how to combine \(y\) from \((y_1, \ldots, y_n)\) so that no \(n - 1\) parties can obtain any information about \(y\)?

**Answer:**

- generalization of \((2, 2)\)
- \(y = y_1 + \ldots + y_n \pmod{q}\)
Assume the case \((n, t)\) for any \(t < n\)

**Question:**

- how to combine \(y\) from \((y_1, ..., y_n)\) so that no \(t - 1\) parties can obtain any information about \(y\)?

**Answer:**

- not so simple anymore
- need *secret sharing*
SHAMIR SECRET SHARING

- Choose random $y_i \in \mathbb{Z}_q$ for all $1 \leq i \leq t$, $t \leq n \leq q - 1$

- Construct a degree-$(t - 1)$ polynomial $f : \mathbb{Z}_q \rightarrow \mathbb{Z}_q$

- s.t. $f(i) = y_i$ for $1 \leq i \leq t$

- Such a polynomial exists and is unique

- Found by using interpolation (again!)

- Send ith secret share $y_i = f(i)$ to party $i \in \{1, \ldots, n\}$

$$f(x) = \sum_{i=1}^{t} y_i \cdot \mathcal{L}_i^S(x) \quad \text{for} \quad S = \{x_i\} \quad \text{and} \quad \mathcal{L}_i^S(x) := \prod_{j=1, j \neq i}^{t} \frac{x - x_j}{x_i - x_j}$$

$$\sum y_i \mathcal{L}_i^S(x_j) = y_j$$

= 1, $x = x_i$
= 0, $x = x_j$ for $j \neq i$
**QUIZ: WHAT IS THE SECRET?**

- *(Answer:) secret: \( y := f(\circ) = \sum y_i \ell_i(\circ) \)*

- **Given \( f(i) \) for any \( t \) inputs \( 1 \leq i \leq n \):**
  - since \( f \) has degree \(-(t-1)\), can recover \( f \) by using interpolation
  - ... thus can recover \( f(j) \) for any other input, including \( f(\circ) \)

- **Given \( f(i) \) for any \( t-1 \) inputs \( 1 \leq i \leq n \):**
  - for *any* \( s' \), there exists a unique \( t-1 \) degree polynomial \( g \) that agrees with \( f(i) \) on these \( t-1 \) inputs and has \( g(\circ) = s' \)
  - \( g \) is obtained from interpolating \((s', ..., f(i), ...)\) at \((\circ, ..., i,...)\)

Note: instead of \( \circ \), one can use any point *not* in \( \{1, ..., n\} \)
**THRESHOLD (N,T)-ELGAMAL**

For $i = 0$ to $t - 1$: $f_i \leftarrow \mathbb{Z}_q$

$f(x) \leftarrow \sum_{i \in [0, t-1]} f_i x^i$

For $i = 1$ to $n$: $y_i \leftarrow f(i)$

$y \leftarrow f(0)$

$pk = h \leftarrow g^y$
Threshold $(N,T)$-ElGamal

\[ h_1 = g^{y_1}, y_1 \]

\[ h_2 = g^{y_2}, y_2 \]

\[ h_3 = g^{y_3}, y_3 \]

\[ h_4 = g^{y_4}, y_4 \]

\[ c = (c_1, c_2) = \text{Enc}(m; r) = (g^{mh^r}, g^r) \]

\[ d_i = \text{Dec}^*(y_i, c) = c_2y_i \]

\[ d_3 = \ldots = c_2y_3 \]

\[ d_4 = c_2y_4 \]

\[ \text{Dec}(\{y_i\}, c) = \log \left( c_1 \prod d_i^{y_i(0)} \right) \]

Note: $\text{Dec}^*$ is the same function as in the last lecture.
THR-ELGAMAL: CORRECTNESS

\[ f(x) = \sum_{i=1}^{t} y_i \cdot \ell^S_i(x) \quad \text{for} \quad S = \{x_i\} \quad \text{and} \quad \ell^S_i(x) := \prod_{j=1, j \neq i}^{t} \frac{x - x_j}{x_i - x_j} \]

- \textbf{Enc} \ (m; r) = (g^{mh}r, gr), \ d_i = c_2 y_i \ \text{for} \ i \in \{x_1, ..., x_t\}
  - = g^{ry_i}

- \prod_i d_i^{l_i(o)} = \prod_i g^{ry_i l_i(o)} = g^{r \sum_i y_i l_i(o)} = g^{rf(o)} = g^{ry} = hr

- \textbf{Thus} \ \log (c_1 / \prod d_i^{l_i(o)}) = \log (g^{mh}r / hr) = m
Remarks

- Dealing can be done without a trusted third party
  - beyond this lecture course
- If users are semihonest, can take any $n \geq t$
- For decryption, at least $t$ must be present
- Also works when a minority of users is malicious
  - then $n \geq 2t + 1$ (beyond this lecture course)

Choice of $t$ depends on application: sometimes it is good to be able to threshold-decrypt with small $t$, but then need each coalition of $t$ parties to only decrypt "allowed values" (i.e., be semihonest)
INFORMATION-THEORETIC MPC

- Everything up to now: only computationally secure
  - secure only under some comput. assumption
  - (except one-time pad)

- Important:
  - one can avoid this in the honest majority model

- **Cost**: additional trust (majority is honest)

The famous “BGW" protocol. See slides of Benny Pinkas at http://u.cs.biu.ac.il/~lindell/mpcschool.html for more information
RECALL: IDEAL MODEL

\[ a \in S_1 \]

\[ b \in S_2 \]

\[ f_a(a, b) \]

\[ f_b(a, b) \]

Trusted third party
**SEMIHONEST REAL MODEL**

**Goal:** (under a cryptographic assumption) semihonest parties should not gain more information than needed.
MPC MODEL

No information is leaked, given a majority of parties is honest

All $n$ parties can have an input and an output

1. Share your input between others
2. Perform computation on shares
3. Send shares of outputs to others
4. Reconstruct your output
REMARKS

- **Ideal model**
  - assume exists a trusted party
  - and secure communication channels...
  - everything is trivially secure

- **Real model**
  - trust nobody but yourself
  - need cryptographic assumptions to get security

**Majority-honest model:**
trust that a *majority* of parties is honest. Not as trivial as ideal model, but can still achieve security without assumptions
STEP 1: INPUT SHARING

Notation: $[a]_i$ is $i$'s share of $a$

$\text{Inp: } s_1$
$\{[s_i]_1\}_i$

$\text{Inp: } s_3$
$\{[s_i]_3\}_i$

$\text{Inp: } s_2$
$\{[s_i]_2\}_i$

$\text{Inp: } s_n$
$\{[s_i]_n\}_i$

$\{[s_i]_j\}_j \leftarrow \text{share } (i, s_i)$
STEP 2: EVALUATION OF CIRCUIT

for each gate, from bottom to top:
if ADD($a_i$, $b_i$, $c_i$) gate: $c_i \leftarrow [a_i] + [b_i]$
if MULT($a_i$, $b_i$, $c_i$) gate: $c_i \leftarrow \text{mult}(i, [a_i], [b_i])$
STEP 3: RECOVERING OUTPUTS

\( \text{s}_1, \{y_j\}_1^i \)
\( \text{s}_2, \{y_j\}_2^i \)
\( \text{s}_3, \{y_j\}_3^i \)
\( \text{s}_n, \{y_j\}_n^i \)

// \text{i} does:
For each \( j \neq i \): send \([y_j]_i\) to \( j \)
Wait until
has received \([y_i]_j\) from at least \( t-1 \) parties \( j \)
Reconstruct \( y_i \) (by using interpolation)
**INPUT SHARING**

**Input:** $s_1$

**Input:** $s_3$

**Input:** $s_2$

**Input:** $s_n$

---

**function share ($i$, $s_i$):**

For $j = 1$ to $t - 1$:

$f_{ij} \leftarrow \mathbb{Z}_q$

$f_{i0} \leftarrow s_i$

$f_i(x) \leftarrow \sum_{j \in [0, t - 1]} f_{ij} \cdot x^j$

// $f_i(0) = s_i$

For all $j = 1$ to $n$: evaluate $f_i(j)$

For all $j = 1$ to $n$: $[s_i]_j \leftarrow f_i(j)$
INPUT SHARING

Input: \( s_1 \)

Input: \( s_2 \)

Input: \( s_3 \)

Input: \( s_n \)

share \((1, s_1)\)

share \((2, s_2)\)

share \((3, s_3)\)

share \((n, s_n)\)
WHY IT WORKS

- For each $i$:
  - Each party $j \neq i$ gets value $[s_i]_j = f_i(j)$
  - Since $\deg f = t - 1$, any $t$ parties can reconstruct $s_i$
    - by pooling their shares and using interpolation
  - No $t - 1$ parties (not involving $i$) can reconstruct $s_i$
EFFICIENCY: INPUT SHARING

- **Computation:**
  - multipoint evaluation: $n$ polynomial evaluations to compute all $f_i(j)$
  - $\Theta(n^2)$ multi-s in $\mathbb{Z}_q$ by using "obvious" algorithm
  
  Special "multipoint evaluation" algorithm: $\Theta(n \log n)$ multi-s in $\mathbb{Z}_q$

function share $(i, s_i)$:
  For $j = 1$ to $t - 1$: $f_{ij} \leftarrow \mathbb{Z}_q$
  $f_{i0} \leftarrow s_i$
  $f_i(x) \leftarrow \sum_{j \in [0, t - 1]} f_{ij} x^j$
  // $f_i(0) = s_i$
  For all $j$: evaluate $f_i(j)$
  For all $j$: $[s_i]_j \leftarrow f_i(j)$

does not really matter: in most applications, $n$ is very small
EFFICIENCY OF SHARING

- **Communication:**
  - each party sends and receives \((n - 1) \log q\) bits
  - Total comm.: \((n - 1)n \log q\)

- **Rounds:**
  - 1 (everything in parallel; no delay)

We will not deal with the question of the size of \(q\); \(q\) is approximately lower-bound by \(n + 1\) and the "data size". In many applications \(\log q = 20\) might be ok.
PRIVATE ADDITION

❖ **Input:** party $i$ has shares $[a]_i$, $[b]_i$ of $a$ and $b$

❖ **Output:** party $i$ has share $[c]_i = [a]_i + [b]_i$ of $c = a + b$

❖ **Needed:**

❖ generate a random degree-$(t-1)$ polynomial $C$, with restriction that $C(0) = A(0) + B(0)$

❖ **Quiz:** how?
**PRIVATE ADDITION**

- **Need:** generate a random degree-$(t - 1)$ polynomial $C$, with restriction that $C(0) = A(0) + B(0)$

- **Solution:** define $C(x) = A(x) + B(x)$
  
  - $C(0) = A(0) + B(0)$
  
  - $C(i) = [c]_i = [a]_i + [b]_i = A(i) + B(i)$

  - $C(i)$ is random if at least one share is random

  Thus if input shares are random, then output shares are random

- Party $i$ locally computes $[c]_i$ by doing 1 addition
PRIVATE MULTIPLICATION

- **Input:** party $i$ has shares $[a]_i = A(i)$, $[b]_i = B(i)$ of $a$ and $b$

- **Output:** party $i$ has share $[c]_i$ of $c = ab$

- **Needed:** generate polynomial $C$ of degree $t - 1$ which is random, except $C(0) = A(0) \cdot B(0)$

- **Quiz:** define $C(x) = A(x) \cdot B(x)$ --- is it ok?
  - partly: $C(0) = A(0) \cdot B(0)$
  - **no:** $C$ is not random, and it also has degree up to $2t - 2$
PRIVATE MULTIPLICATION

Recall: \( C(x) = A(x) B(x) \)
- not random, has degree \( 2t - 2 \)

Need: \( 2t - 2 \leq n \)

also: if \( n/2 \) parties pool their data, they get a degree \( n \) polynomial \( \Rightarrow \) hence \( t < n/2 \)
PRIVATE MULTIPLICATION

- **Idea:** party $i$ has $[a]_i[b]_i = C(i)$
- He creates a random polynomial $D_i$ of degree $t - 1$, s.t. $D_i(0) = [a]_i[b]_i$
- ... and shares it
- Denote $D(x) := \sum_{i \in [1,n]} D_i(x) l_i(0)$
- $D(0) = \sum_i D_i(0) l_i(0) = \sum [a]_i[b]_i l_i(0)$
  $= \sum C(i) l_i(0) = C(0) = ab$ (secret)
- The new share of $ab$ is $[ab]_i = \sum_{j \in [1,n]} D_j(i) l_j(0) = g(i)$

function `mult (i, a_i, b_i)`:
For $j = 1$ to $t - 1$: $D_{ij} \leftarrow \mathbb{Z}_q$
$D_{i0} \leftarrow [a]_i \cdot [b]_j$
$D_i(x) \leftarrow \sum_{j \in [0,t-1]} g_{ij} x^j$
// $g_i(0) = [a]_i \cdot [b]_i$
For all $j$: evaluate $D_i(j)$
For $j \neq i$: send $D_i(j)$ to $j$
Wait until received $D_j(i)$ from all $j \neq i$
Set $[ab]_i \leftarrow \sum_j D_j(i) l_j(0)$

$D(X)$ is a random polynomial, not known by any single party, s.t. $D(0) = ab$
EFFICIENCY PER PARTY

- **Computation:**
  - dominated by multipoint evaluation and interpolation
  - Computing **all** \( l_j(\circ) \)
  - \( \Theta(n \log n) \mod q \) multiplications

- **Communication:**
  - \((n - 1) \log q\) bits per party

- **Rounds:** 1

---

function mult \((i, a_i, b_i)\):

- For \( j = 1 \) to \( t - 1 \):
  - \( D_{ij} \leftarrow \mathbb{Z}_q \)
  - \( D_{io} \leftarrow [a]_i \cdot [b]_j \)
  - \( D_i(x) \leftarrow \sum_{j \in [0, t-1]} g_{ij} x^j \)
  - // \( g_i(\circ) = [a]_i \cdot [b]_i \)

**For all \( j \): evaluate** \( D_i(j) \)

- For \( j \neq i \): send \( D_i(j) \) to \( j \)

Wait until received \( D_j(i) \) from all \( j \neq i \)

- Set \([ab]_i \leftarrow \sum_j D_j(i) \leftarrow l_j(\circ) \)
RECONSTRUCTION OF SECRETS

- For each party $i$ with private output $y_i$:
  - Obtain at least $t - 1$ shares $[y_i]_j$ of $y_i$ from other parties
  - Use interpolation to compute $y$ from them
SECURITY

- Every set of \( t \)-1 players receives in each round values that are \((t-1)\)-wise independent, and thus uniformly distributed. \((t-1)\)-wise independence means intuitively: each \((t-1)\)-dimensional subvector of the full vector \( (x_1, \ldots, x_n) \) looks random.

- Hence no information about the actual wire values is leaked.
Let $M$ be the number of multiplication gates

**Computation:**
- Dominated by appr. $M$ multipoint evaluations / interpolations
- $\Theta (Mn \log n)$ operations in $\mathbb{Z}_q$

**Communication:**
- Approx. $Mn$ messages or $Mn \log q$ bits per party

**Rounds:**
- multiplicative depth of circuit
# COMPARISON

<table>
<thead>
<tr>
<th></th>
<th>Rounds</th>
<th>Communication</th>
<th>Computation</th>
<th>Trust</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDD-based</td>
<td>🍒👍</td>
<td>🍒👍</td>
<td>🍒👎</td>
<td>Comp. assmpt.</td>
</tr>
<tr>
<td>Comp. MPC</td>
<td>🍒👎</td>
<td>🍒👎</td>
<td>🍒👍</td>
<td>Comp. assmpt.</td>
</tr>
<tr>
<td>Inf theor. MPC</td>
<td>🍒👎</td>
<td>🍒👎</td>
<td>🍒👍</td>
<td>&quot;majority are honest&quot;</td>
</tr>
</tbody>
</table>
TRADEOFFS

✧ **Efficiency:**
  ✧ More efficient with a smaller number $n$ of parties

✧ **Trust:**
  ✧ With larger $n$, tolerance against a larger number $t$ of corrupted parties

✧ In practice, $n$ depends on application:
  ✧ how many parties are needed, how many applicable, how many we can trust...?
NEXT LECTURE

- Garbled circuits
  - very good computation, rounds
  - bad communication