Advanced Tools from Modern Cryptography

Lecture 13

MPC: Honest-Majority + Active Corruption

UC-Secure Information-Theoretic MPC

- UC secure MPC protocols for general functions
- UC security without honest-majority
  - Needs setup (e.g., GMW paradigm, using CRS for ZK)
  - In fact, information-theoretic security possible, given OT
- UC security with honest Majority:
  - No setup needed
  - With selective abort if < n/2 parties corrupt
  - Can even get guaranteed output delivery and perfect security if < n/3 corrupt: **BGW Protocol** (Today)

Verifiable Protocol Execution

- We already saw passive secure BGW protocol
- So need to only implement a functionality F_{VPE} which carries out the protocol on behalf of all the parties
  - Progress? Seems like we still need MPC for general functions!
    - But easier: Every variable/computation in F_{VPE} is "owned" by some party

VPE Functionality

- F_{VPE} maintains a state for each party (image), and carries out "public" instructions (sent by a majority of parties) on these images
- F_{VPE} supports:
  - Uploading a variable to one's own image. The value being uploaded is private. (The operation itself is public.)
  - An addition or multiplication within an image
  - Transferring a variable from one image to another
  - Can at any point read a variable in one's own image
- Plan for implementing F_{VPE}: Every variable will be maintained as a <u>commitment</u> by its owner to the others

Commitment: First Cut

- Simply do (n,t+1) secret-sharing of the message among all the n players (e.g., degree t Shamir secret-sharing)
  - To reveal, sender broadcasts all the shares and all the parties must agree. If the broadcast shares are valid, accept reconstruction. Else abort.
  - For $n-t \geq t+1$ (i.e., $t < n/2$), honest parties' shares already define a unique secret. Corrupt sender (in a collusion of t players) cannot open to two values
- Problem 1: A single corrupt party can cause abort
- Problem 2: Does not ensure that there is a valid commitment! If commitments are not just opened, but computed on, problematic.

Commitment with Guaranteed Opening

- When $t < n/3$, can prevent adversary from causing abort at any point (except, a corrupt sender can make all honest parties abort)
- Idea: Before accepting a commitment, do consistency checks to ensure that honest players' shares do define a valid polynomial.
  - Problem: Corrupt parties can claim inconsistency with honest players' shares ("dispute")
  - Idea: Let sender resolve disputes between two parties by publishing both their shares
  - Problem: Adversary sees more information by disputing.
  - Idea: Information published is already known to the adversary

Commitment with Guaranteed Opening

- Committing: Use a bivariate polynomial $f(x,y)$, of degree t in each variable, with $f(0,0)$ being the message. Party P_j gets $f(i,j)$ for all i.
  - i.e., Party P_j gets a degree t univariate polynomial $f_j(x) := f(x,j)$
  - Will require $f(i,j) = f(j,i)$

$f(x,y) = \Sigma\, c_{p,q}\, x^p y^q$, with $c_{p,q} = c_{q,p}$ and $c_{0,0}$=msg

- Checking:
  - P_i and P_j check if $f(i,j) = f(j,i)$
  - Also, P_j checks what it got is indeed a degree t polynomial
- Disputing: If either check fails, P_j broadcasts a complaint
  - Resolution: Sender broadcasts $f(i,j)$ or degree-t f_j respectively
- Repeat until no more disputes
- If sender caught cheating in its broadcast, all honest parties abort

Commitment with Guaranteed Opening

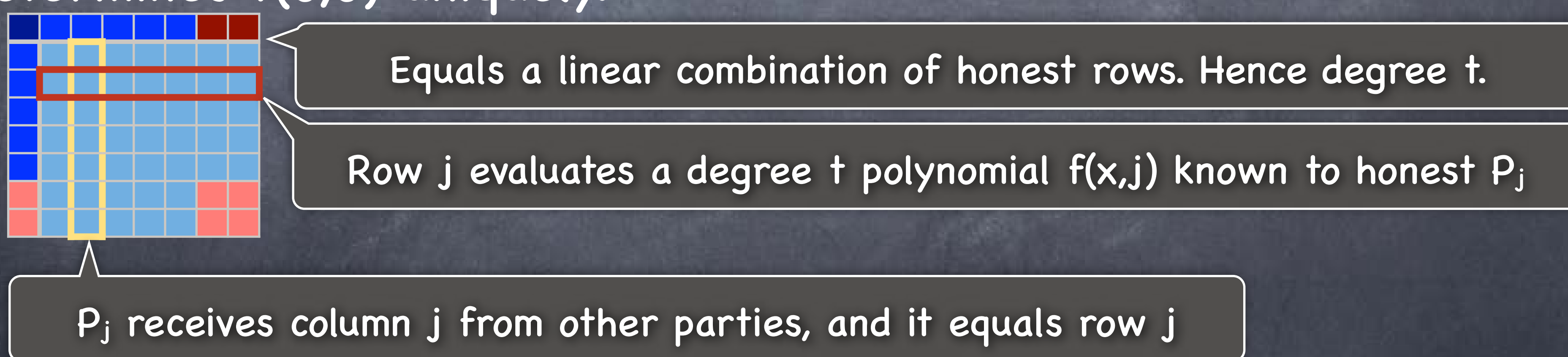
- If sender honest
  - Before any disputes, corrupt players (<t) learn nothing about the message
    - There is a bijection between sharings of m and sharings of 0, which preserves the view of the adversary

      Also, each sharing is equally likely
      - Consider degree t polynomial $h(x)$ s.t. $h(0)=1$, and $h(j)=0$ for all corrupt P_j
      - Bijection maps $f(x,y)$ to $f(x,y) - m \cdot h(x)h(y)$
  - Messages revealed during dispute resolution are all messages known to the corrupt parties
- Opening: Each party P_j computes and sends $f(0,j)$ to the receiver. Receiver reconstructs the degree t polynomial $f(0,y)$, with error correction from up to t errors [algorithm omitted]

  Not relying on sender

Commitment with Guaranteed Opening

- If sender corrupt:
  - Either sender aborts before all disputes settled,
  - Or, no dispute remaining among the honest players. Then $\{ f(i,j) \mid i,j \text{ honest} \}$ is part of a valid sharing of $f(0,0)$, and determines $f(0,0)$ uniquely.



  - **Opening:** Each party P_j computes and sends $f(0,j)$ to the receiver. Receiver reconstructs the degree t polynomial $f(0,y)$, <u>with error correction from up to t errors</u> [algorithm omitted]

Why $t < n/3$?

- $t<n/3$ needed for broadcast with guaranteed output delivery (later)
- Even if broadcast given as an ideal functionality, the BGW protocol needs $t < n/3$
  - To uniquely decode a codeword from $\leq t$ errors, need distance between valid codewords to be $> 2t$ (otherwise can have an invalid codeword which is t away from two valid codewords). But for degree t polynomials, minimum distance $= n-t$ [Why?].
    So, $n-t > 2t$. i.e., $n > 3t$
- Note: Given broadcast, there are protocols that can tolerate $t < n/2$ corruption with statistical security (BGW has perfect security)

Recall VPE Functionality

- F_{VPE} maintains a state for each party (image), and carries out "public" instructions (sent by a majority of parties) on these images
- F_{VPE} supports:
  - Uploading a variable to one's own image. The value being uploaded is private. (The operation itself is public.)
  - An addition or multiplication within an image
  - Transferring a variable from one image to another
  - Can at any point read a variable in one's own image
- Plan for implementing F_{VPE}: Every variable will be maintained as a commitment by its owner to the others

A VPE Protocol

- Every variable maintained as a commitment by its owner to the others

Commitment is using the symmetric bivariate polynomial secret-sharing.

- Uploading: Commitment
- Linear operations: If f, g shares of a, b, then $\alpha f+\beta g$ is a share of $\alpha a+\beta b$ (with the same dealer)

For guaranteed output, if a party doesn't make a commitment, open up its entire image

- Multiplication: Owner should send a fresh commitment of c and give a proof of $c=a\cdot b$, that can be verified collectively
  - Proof of $c=a\cdot b$: Pick degree t polynomials p, q with constant terms a, b, and let r=p.q, a degree 2t polynomial with constant term c. a,b,c already committed. Commit other coefficients. Evaluations p(i), q(i), r(i) are computed (using linear operations) and revealed to party P_i who checks if $p(i)\cdot q(i) = r(i)$. If all $n-t > 2t$ honest parties agree, then indeed $p\cdot q=r$.

A VPE Protocol

- Every variable maintained as a commitment by its owner to the others

Commitment is using the symmetric bivariate polynomial secret-sharing.

- **Uploading:** Commitment
- **Linear operations:** If f, g shares of a, b, then $\alpha f+\beta g$ is a share of $\alpha a+\beta b$ (with the same dealer)

For guaranteed output, if a party doesn't make a commitment, open up its entire image

- **Multiplication:** Owner should send a fresh commitment of c and give a proof of $c=a\cdot b$, that can be verified collectively
- **Transfer:** To transfer a committed variable a from P_i to P_j, a is opened to P_j and P_j recommits it. Then P_j proves equality
  - To prove committed values a, b are equal, P_j commits to coefficients of (identical) degree t polynomials p, q with constant terms a, b respectively, and p(k),q(k) opened to P_k who checks p(k)=q(k)

Use original commitments

Linear combination of coefficients

Broadcast

- Our protocol relied on broadcast to ensure all honest parties have the same view of disputes, resolution etc.
- Concern addressed by broadcast: a corrupt sender can send different values to different honest parties
- Broadcast with selective abort can be implemented easily, even without honest majority
  - Sender sends message to everyone. Every party cross-checks with everyone else, and aborts if there is any inconsistency.
- If corruption threshold $t < n/3$, then it turns out that broadcast with guaranteed output delivery can be implemented [omitted]
- If broadcast given as a setup, can do MPC with guaranteed output delivery for up to $t < n/2$