Lesson 18 Lecture Notes

CS555

1 Secure Multi-party Communication

- Multiple parties each with their own keys.
- Together they can compute some overall key with f(keys), but apart they cannot.
  - Example: shared key is the average of all parties' keys.
- Intermediate communications should be zero-knowledge.
  - Only leakage is the final output.

1.1 Properties

1. **Utility / Correctness:** When needed parties get together, it is always possible to obtain the correct output.
2. **Security:** All except the final output maintain security. Intermediates are indistinguishable w.r.t. inputs.

1.2 Leakage

- Two types of leakage:
  1. **Intermediate** (internal communications within parties) $\rightarrow$ can be secured.
  2. **Output** (public communications) $\rightarrow$ guaranteed leakage.
     - Add noise to the output to help hide leakage.

2 Noise

- Noise should be on the magnitude of the output itself.
- Adding multiple parties' noises $\rightarrow$ noise compounds.
- **Solution to compounding noise:** MPC
  - Does not require a true central trusted mediator $\rightarrow$ securely simulate a virtual party that collects info and outputs f(info), where only f is leaked.
  - Mitigates trust gap between centralized and localized computation.
  - If you can securely gather all party info and compute on it, only noise at the end is required $\rightarrow$ increased utility/correctness.

2.1 Assumptions

- **Honest majority assumption:** all users follow protocol.
  - Information-theoretically secure protocols exist for n/a corruptions.
- Corrupted parties can collude or lie.
  - **Utility:** if malicious users can lie, correctness of output cannot be guaranteed.
  - **Security:** still possible even with malicious users.
- **Byzantine Robustness:** With 3 inputs, even if 1/3 is arbitrary, the output should still be close to the true value.

2.2 What about $n-1$ of n corrupted?

Goldreich-Micali-Wigderson Protocol

- Uses Oblivious Transfer.
- Only computationally secure.
- **Example:** Average Salary in a Room
  - Each person i has salary S_i and passes Y_i to person Y_{i+1} by the following equation where R is a random number only known by Bob (person 1)

$$Y_1 = S_1 + R$$
$$Y_2 = Y_1 + S_2$$
$$\vdots$$
$$Y_n = \sum_{i=1}^{n} S_i + R$$

Bob knows R, so he can subtract:

$$Y_n - R = \sum_{i=1}^{n} S_i \pmod p$$

- having one corrupted party $\rightarrow$ still secure
- With two malicious users: can learn salary of victim sandwiched between them.
  - Eve 1 passes $Y_1 = 0$; victim passes to Eve 2 $Y_2 = S_2 + Y_1 = S_2$

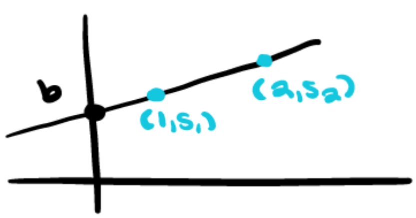
3 Key Tool: Secret Sharing

- Dealer has secret b.
- Wants to split amongst n users so that only when some t users come together they can have full secret.
- Any "authorized" subset can recover b.

3.1 Types

- **n-out-of-n:** Each share looks random, but sum of all shares is b.
  - $s_1, \dots, s_{n-1}$ chosen random.
  - $s_n = b - (s_1 + s_2 + \cdots + s_{n-1}) \pmod p$.
  - Any subset without s_n learns nothing, any $(n-1)$ or less with s_n can't determine b because missing some random secret share
- **1-out-of-n:** Everybody gets b.
- **2-out-of-n:** generate i random numbers; then person i gets a vector with difference $b - r_j$ for all $j \neq i$
- **t-out-of-n:** for all subsets of size t containing person i add to their secret share $b-$(sum of subset randoms)
  - Costly: requires (n choose t-1) secret keys per person.
  - Efficient in updating shares.

3.2 Shamir's t-out-of-n Secret Sharing



- Dealer defines random polynomial $f(x) = a_0 + a_1x + \cdots + a_{t-1}x^{t-1} + b$.
- Each participant receives a point (i, s_i) lying on $f(x)$.
- Any t points can interpolate polynomial and recover b.
- Solution given by Lagrange interpolation.