MPC optimizations

Common MPC optimizations

- Offline/online model
- Precomputation in plaintext
- Hybrid MPC protocol

Additive secret sharing

- $\langle a \rangle = \{a_0, a_1\}$ indicates an additive secret sharing such that $a = a_0 + a_1 \mod 2^l$
- Addition: $\langle c \rangle_i = \langle a \rangle_i + \langle b \rangle_i$
- Multiplication requires interaction, and some heavy weight crypto:
  - $c = a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1$, but only two partial products can be computed locally
  - Can use HE/OT to compute shares of the other two partial products
    - P_0 encrypts a_0, sends to P_1; P_1 computes Enc$(a_0b_1 + r)$, sends to P_0; P_0 decrypts ciphertext

Offline/online: Beaver's triples

- If parties already have $\langle u \rangle, \langle v \rangle, \langle z \rangle$, where u, v are uniformly random, and $z = uv \mod 2^l$, then can easily use this triple to compute multiplication without heavier crypto like HE
    - P_i computes $\langle e \rangle_i = \langle a \rangle_i - \langle u \rangle_i, \langle f \rangle_i = \langle b \rangle_i - \langle v \rangle_i$
    - Both parties reconstruct e and f
    - $\langle c \rangle_0 = f \cdot a_0 + e \cdot b_0 + z_0$
    - $\langle c \rangle_1 = -e \cdot f + f \cdot a_1 + e \cdot b_1 + z_1$

Offline/online: Beaver's triples

- A reformulation of multiplication
- Split normal MPC into input independent phase & input dependent phase
- Random triples can be generated in advance, without knowing the inputs
- Offline (not quite): parties use heavy crypto (HE, OT) to compute random triples
- Online: parties use light weight crypto to execute the computation, given inputs
- Good for any interactive computation that can have a preprocessing phase

Plaintext precomputation

- Why do all of the computation in MPC, if some of it can be (securely) done in plaintext?
- Simple example: sort every party's input
  - Each party can pre-sort its own input in plaintext; MPC can merge
- Harder example: linear regression without SGD
  - Each party can precompute $X_i^T X_i$, then sum it up in MPC, followed by a global matrix inverse computation

Hybrid MPC protocol

- No single MPC primitive wins for all workloads
- Circuit type is different: arithmetic is good for representing matrix multiplication, boolean is good for comparisons (e.g., ReLU)
- Different performance characteristic under a different number of parties, different network (LAN/WAN), memory constraints
    - GC circuits are very gigantic due to the labels & gate encryption
    - Secret sharing could have multiple round trips

Hybrid MPC protocol

- ABY (Arithmetic, Boolean, Yao)
- A general, two-party framework for converting between A, B, and Y
- Example: Arithmetic to Yao conversion (A2Y)
    - P_0 is the garbler, P_1 is the evaluator
    - Parties have arithmetic shares $\langle x \rangle^A = \{x_0, x_1\}$
    - Secret share each arithmetic share as a Yao share
        - P_0 creates labels for x_0 and sends P_1 the correct labels
        - P_0 also creates labels for x_1, and P_1 runs OT to get the correct labels
    - Reconstruct x inside GC using an addition circuit

Today’s reading: CNN inference using Gazelle

Next class: MPC with malicious security

- So far, all of the protocols have been constructed with semihonest adversary
- What can a malicious adversary do?
    - Give fake inputs (not something MPC can handle)
    - Give inconsistent inputs
    - Execute incorrect computation
    - Refuse to release the final result (cannot always be handled)
- Optimizations are tricky!
    - Sorting optimization?