CMSC 858K — Introduction to Secure Computation October 9, 2013

Lecture 15

Lecturer: Jonathan Katz *Scribe(s): Adam Groce*

1 Semi-honest MPC implementations

1. Fairplay MP
   - Implements Beaver-Micali-Rogaway with BGW for the outer protocol (so $t < n/2$).
   - 3 types of parties: input, output, and computation
     - These can overlap (simple setting - all parties are all three)
     - Use 3 computation nodes - this represents MPC being provided as a service. Each computation node would be run by a different company.
   - Experiments
     - 5 to 11 parties considered.
     - Up to 2^10 gates.
     - 5 parties and 2^10 gates took about 10 seconds
     - A second-price auction with a seller, 4 bidders, and 8-bit bids can be done with 400 gates.
2. Danish beet auction
   - One major point: lots of real-world tasks can be done with adding, multiplying, and comparing 32/64-bit integers.
   - Major goal: trying to get simple arithmetic circuits for comparison.
   - Task had many input providers (1000 farmers)
   - Three computation nodes (farmers' organization, clearinghouse company, and researchers)
   - Each farmer specifies at each potential market price whether they would like to buy or sell and the quantity.
   - Functionality computes the market-clearing price.
3. Secure SCM
   - MPC applied to supply chain management.
   - Distribution schedules require coordination between many suppliers that don't want to share details of their operations.
4. SEPIA

- Uses BGW, $t < n/2$.
- 5 parties run as compute nodes
- 83,000 multiplication gates per second (Fairplay MP gets 2, VIFF gets 300)

5. Choi et al. (at UMD)
   - Uses GMW, boolean circuits

2 Malicious Security Definition

We now define security for the malicious setting, where deviations from the protocol are allowed. In order to do so, we need to define both the real and ideal worlds. Both will be similar to the version defined for semi-honest security. The changes are outlined below.

2.1 Real world

In the real world, we now allow malicious parties to deviate arbitrarily from the protocol. This includes changing their input, or behaving in ways that are not clearly associated to any particular input value.

There are two potential models of communication. One is the asynchronous model, where parties wait indefinitely for the next message from another party. The alternative, which is what we will study, is the synchronous model. In this model, we assume all messages are sent within some fixed time. This means that a lack of message by the cutoff time will be interpreted as a null message. We will, however, assume that adversaries are "rushing", meaning that in a given round they receive all messages from honest parties before computing and sending their own message(s).

We also assume a broadcast channel, meaning that a given party can send a message in a way that guarantees the same message is received by everyone else, even if the sending party is dishonest. This can be achieved in two ways. One is to have a physical broadcast channel, for example wireless transmission with nearby receivers. Alternatively, a protocol can be used that simulates a broadcast channel using only point-to-point channels. (Formally, the security requirement is that all honest parties agree on the output, and if the sender is honest that output matches the sender's input.) Broadcast protocols are, however, expensive to run.

2.2 Ideal world

We also modify the ideal world to allow malicious parties to change inputs. (No protocol could prevent parties from using the "wrong" input.) We now divide the definition into two cases, depending on whether "fairness" is required. It is always possible for some parties to refuse to participate and therefore prevent any output from occuring. Fairness says that deviation or an early abort by some parties must deny outputs to either everyone or no one - it cannot be that some but not all parties get output. Unfortunately, fairness cannot be guaranteed in general when $t \geq n/2$. We therefore create two definitions, one that requires fairness and one that does not.

In "full security" fairness is required. The honest parties all send their real input to the functionality F, while malicious parties send anything. F then computes $(y_1, \ldots, y_n) = F(x_1, \ldots, x_n)$ and sends y_i to P_i.

The alternative is "security with abort". Here things begin as before, but after F computes the output, it sends y_i values only to malicious players. The attacker then chooses to abort or continue. If the attacker chooses to continue, honest players are also sent their output. If the attacker aborts, they instead receive an error symbol $\perp$.