CSE 5854: Class 15

Benjamin Fuller

March 20, 2018

1 A semi-honest multi-party computation protocol

Before break we described a semi-honest protocol to compute any function in polynomial time. The protocol was split into three phases (consider a computation represented as a circuit). The three phases are:

1. Sharing each input to the computation x_i to each party. Call these shares $x_{i,1}, ..., x_{i,n}$.
2. Have parties compute each gate on the shared input values to the gate. We perform these computations doing the following:
   (a) NOT gate: On input $x_1, ..., x_n$ the first party (arbitrarily selected) inverts their share.
   (b) XOR gate: On input x_i, y_i each party locally adds the shares to produce a share of the output value.
   (c) AND gate: This protocol was more complicated and relied on a party generating a new random value and then running a 1-out-of-4 oblivious transfer.
3. Reconstruct the final output by transmitting the shares, y_i, of the output to each party.

This forces the core of the semi-honest GMW protocol. In this class we will talk about two ways to make this protocol withstand malicious behavior.

1. A protocol that is secure when at most $t < n$ parties are malicious. This protocol makes no attempt to provide fairness.
2. A protocol that is secure and fair when at most $t < n/2$ parties are malicious.

Note It is impossible to provide fairness if at least half of the parties are malicious. Roughly, you can think of the impossibility as follows: suppose such a protocol exists. Two players P_1 and P_2 emulate a two party computation by running the n party computation with each player controlling $n/2$ parties. If the n party protocol is fair then we know that neither P_1 or P_2 can abort and break fairness. However, we know that such a protocol cannot exist thus the n party protocol must not be secure.

The idea behind fairness is relatively simple in the semi-honest model. We ask each party to secret share their input using an $n/2$-out-of-n secret sharing as the first message in the protocol. If honest parties detect that a party has stopped responding they can get together to construct the input of that party. This discussion assumes a synchronous network where the adversary is not allowed to drop messages. If the adversary controls the network or the network is asynchronous it is not possible to distinguish between a party not participating (in which case their input should be opened) and the adversary blocking a message (in which case their input should be kept private). Thus, if an adversary can drop messages we do not attempt to provide fairness.

Both of these protocols can be transformed from semi-honest to malicious behavior.

1.1 The rough idea

In the two party setting we saw a general paradigm for converting semi-honest protocols to malicious protocols. Let H_1 and H_2 be the code of the honest party. It worked as follows:

1. Each party commits to their inputs.
2. Party P_1 and P_2 conduct coin flipping protocols. Specifically we consider a single party P_1. At the end of the protocol P_1 has committed to r_1 and P_2 has sent r_2. The random tape to be used by P_1 is $r_1 \oplus r_2$. Importantly, P_1 never opens its commitment. Instead, it will prove that its messages are consistent with $r_1 \oplus r_2$.
3. For each message m to be sent by P_1 party sends the message and proves (using ZKPoK) that $m \leftarrow H_1(\sigma; r_1 \oplus r_2, x_1)$ is the message the honest party would send on transcript σ with randomness $r_1 \oplus r_2$ on input x_1. Note that this function is well defined, the party is deterministic based on its randomness, input, and messages received. The witness for the proof is the randomness $r_1 \oplus r_2$ and the input. Further note that these two values must be private and so can't be part of the statement. However, they were both "committed" to at the start of the protocol so P_1 can prove they are being consistent with that initial commitment. The statement for the language is the starting commitments and the current transcript.

In order to be able to force honest behavior we had to make the honest behavior deterministic. The three things that should effect the honest behavior are the input, the randomness, and the messages received from other parties. We split these three things into two piles: input and randomness are committed to at the beginning but secret, the messages are "public" since there are only two parties.

We stress this separation because it gets significantly more complicated when we move to the n party setting. Consider the simple setting with just three players P_1, P_2 and P_3. We can have parties commit and do a multi party coin flipping protocol. The trouble arises when we try and emulate the honest behavior. Suppose that P_2 is supposed to send a message m to P_3 that depends on the message y it received from P_1. In order for P_2 to prove that m is the honest message P_3 needs to have a full view of the messages received by P_2. There are two issues:

1. How do we communicate this view to P_3. The two obvious choices are for P_2 to announce the message it received from P_1 and for P_1 to announce this message. However, both of these options seem ripe for cheating. If P_2 is announcing the message it might be inconsistent with what they actually received. If P_1 is announcing they can lie about the message they sent to P_2. It is possible to use signatures to prevent this cheating (with quite a bit of care). However, there is still a second problem.

2. What if seeing y breaks privacy? In the case where both P_1 and P_2 are honest. The privacy of the protocol might depend on the two parties having private communication between them. We could force P_1 and P_2 to use encryption and thus it is safe to release the message y.

Both of these problems get more complicated as we add multiple parties. We need to be sure that signatures and encryption are used in the right way. Furthermore, there is a bigger problem. Consider the case where there are four parties $P_1, ..., P_4$. Suppose that P_1 and P_2 are malicious. They can use each other as randomness to set messages in a way that they prefer. So P_2 can ask P_1 for a specific y that allows them to act in a malicious way (or in a way that depends on a received message). Furthermore, these two parties can coordinate and act as though P_1 sent a different message when communicating with P_3 and P_4. The core of our maliciously secure transform was to remove all sources of randomness from the malicious parties and force them to act in a preset way. While it isn't clear how to use this freedom it seems very dangerous.

1.2 The full idea

To deal with this problem in full generality requires quite a lot of overhead. The idea is to make sure that everyone receives every message so that we can properly enforce semi-honest behavior. However, we still need the ability to send a message to only one individual so we will encrypt messages on top of this channel. We transform our protocol in the following two ways:

1. We use point-to-point channels to emulate a broadcast channel where every message is received by every party.

2. We use public key cryptography to emulate private channels on top of the broadcast channel that we are emulating using point to point channels.

The second of these changes is much easier. We add a step to the protocol where every party broadcasts a public key and then all messages are encrypted (using a CCA secure encryption) before being sent on the broadcast channel. Notice this change has a large efficiency hit as we are using broadcast but only sending messages to one recipient. As we'll see creating broadcast has an overhead larger than the number of parties.

1.3 Creating authenticated broadcast

Creating a broadcast channel from point-to-point channels is an area of interest in both cryptography and distributed systems. The general problem is known as either byzantine agreement or byzantine broadcast. In the information-theoretic case it is known that byzantine agreement is not possible if at least one third

of parties behave arbitrarily. Creating a broadcast channel is the focus of the next class.