CMSC 858K — Introduction to Secure Computation October 21, 2013

Lecture 20

Lecturer: Jonathan Katz *Scribe(s): Xiong Fan*

1 Summary

In this lecture, we describe a multiparty computation (with abort) in the malicious setting, assuming the existence of broadcast channel.

2 Two Party Computation

We describe a protocol for two party computation below, which is a special case of multiparty computation without honest majority. Assume the existence of a semi-honest protocol Π_{sh} for P_1, P_2. This is known as the GMW compiler.

Input Commitment : In this step, the two parties exchange the commitments of their input following by a zero-knowledge proof of knowledge of the inputs.

- P_1 sends the commitment of his input $Com(x)$ to P_2.
- P_1 and P_2 engage in a zero-knowledge proof of knowledge protocol, showing that P_1 know the input x and the commitment $Com(x)$.
- P_2 sends the commitment of his input $Com(y)$ to P_1.
- P_1 and P_2 engage in a zero-knowledge proof of knowledge protocol, showing that P_2 know the input y and the commitment $Com(y)$.

Coin Generation : In this step, the two parties engage in secure protocols, which one party receives a commitment to a random string and the other party receives the string itself plus the decommitment of the string.

- P_1 and P_2 engage in a modified coin-tossing protocol. Then P_1 obtains a random string r_1, the commitment and decommitment of the string $Com(r_1), decom(r_1)$, while P_2 obtains the commitment $Com(r_1)$.
- P_1 and P_2 engage in a modified coin-tossing protocol. Then P_2 obtains a random string r_2, the commitment and decommitment of the string $Com(r_2), decom(r_2)$, while P_1 obtains the commitment $Com(r_2)$.

Protocol Emulation : The two parties run the semi-honest protocol Π_{sh} with (x, r_1) and (y, r_2) while proving that their steps are consistent with input string, random tapes and previously received messages in zero knowledge setting.

3 Security Proof

Theorem 1 *If Π_{sh} is semi-honest two party computation, commitment scheme and zero knowledge proof are both secure, then the scheme described above is secure with abort in the $(ZKPoK, cr)$-hybrid model.*

Proof Assume P_2 is malicious. The simulator does the following steps:

- Commit to 0.
- Simulate the output of F_{ZKPoK}.
- Receive $Com(y)$.
  - Extract y from the message P_2 sends to F_{ZKPoK}.
  - Sends y to ideal functionality for f and gets back output z.
- Run simulator for Π_{sh} on (y, z) to get $(r, trans)$.
- $Com(0)$ from F_{ct}.
- Set $r_2 = r$ and give $r_2, Com(r_2), decom(r_2)$ to P_2 as output from F_{ct}.
- Run the end of Π by using messages from $trans$ and giving simulated ZK proofs (verifying the proof of P_2).

We then describe a series of hybrid games to prove the security:

Hybrid 1 : Real execution.

Hybrid 2 : Replace all proofs from P_1 with simulated proofs.

Hybrid 3 : Replace commitments from P_1 with $Com(0)$.

Hybrid 4 : Run Π_{sh} using (x, r_1, y, r_2), where y is extracted as above, to get output z and P_1's message $trans$.

- Use z as the output of P_1.
- Use $trans$ in the last phase of the protocol.

Hybrid 5 : Compute $z = f(x, y)$ and use that though out the protocol.

Hybrid 6 : Replace Π_{sh} with $Sim_{\Pi_{sh}}(y, z)$.

■

Theorem 2 *(Informal) The same approach who achieves security with-abort in the multi-party setting, assuming broadcast is available.*