CMSC 858K — Introduction to Secure Computation	October 25, 2013
Lecture 22	
Lecturer: Jonathan Katz	*Scribe(s): Xiong Fan*

1 Protocols of Authenticated Broadcast

Definition 1 A message is (v, i)-*authentic* for P_k if it has the form $(v, P_s, \sigma_s, P_2, \sigma_2, ..., P_i, \sigma_i)$, such that:

- $\{P_s, P_2, ..., P_i\}$ are distinct and do not include P_k.
- σ_s is a valid signature for pk_s on v.
- σ_j is a valid signature on $(v, \sigma_s, \sigma_2, ..., \sigma_{j-1})$.
- received in round i.

$\diamondsuit$

The protocol for authenticated broadcast is the following:

Round 1 : Party s signs v, and sends (v, P_s, σ_s) to all parties.

Round 2 though $n-1$: If P receives a (v, i)-authentic message in the previous round, it adds its signature and sends $(v, ..., P, \sigma_P)$ to all parties.

Output :

- If it receives a $(0, i)$-authentic message and a $(1, i)$-authentic message, it outputs 0.
- If it receives no authentic message, it outputs 0.
- If it receives only (v, i)-authentic message, it outputs v.

Correctness: It is immediate by the signature scheme.

Agreement: Say honest party P_j get a (v, i)-authentic message. There exist two cases:

- If $i < n-1$, then P_j will add his signature and send a $(v, i+1)$-authentic message to everyone. If P_k had already signed, then he preciously received a (v, i)-authentic message, otherwise you receive a $(v, i+1)$-authentic message.
- if $i = n-1$, then the (v, i)-authentic message includes signatures from all other parties. So any honest party must have received a (v, i)-authentic message.

2 Fully Secure Multiparty Computation with Honest Majority

In this section, we apply the above protocol in the setting of secure multiparty computation. We assume the number of parties and corrupted parties are n, t correspondingly. The scenario can be described in two cases:

- $t < \frac{n}{3}$ in the plain model.
- $t < \frac{n}{3}$ with a broadcast channel or a PKI.

Let Π be a semi-honest protocol. The fully secure protocol is the following:

- Every party commits to its input and give a *ZKPoK* of the input.
- Every party does a *Verifiable Secret Sharing* of its input.
- Parties run coin-tossing function, giving a commitment of a random value held by each party P_i.
- Every party does a *Verifiable Secret Sharing* of its randomness.
- Run protocol Π, giving *ZK*-proof of correctness. If any party cheats, reconstruct their input and randomness and continue the protocol for them.

A *Verifiable Secret Sharing* is a protocol such that:

- P holds x and $decom$.
- Everybody knows $Com(x)$.

Then we describe the verifiable secret sharing protocol:

- P generates shares $(x_1, ..., x_n)$ of x using Shamir secret sharing with threshold $t + 1$.
- P broadcasts $Com(x_1), ..., Com(x_n)$.
- P sends to P_i the value $x_i, decom_i$ by broadcasting these values under P_i public key.
- P gives *ZK*-proof of correctness.