# Layer 3 – Interactive Protocols

**Standard commitment schemes**

1. Pedersen commitments: <http://cs.nyu.edu/courses/fall01/G22.3033-003/lect/lecture14.ps> , Section 2.5

**Trapdoor (equivocal) commitment schemes**

1. Based on DLOG Sigma protocol [HL, Section 6.6, use DLOG sigma as basis]

**Extractable commitment schemes**

**Homomorphic commitment schemes**

**Sigma protocol**

1. Sigma protocol of DLOG [HL, Section 6.1]
2. Sigma protocol of DH tuple [HL, Section 6.2]
3. Template for Sigma protocol – programmer fills in procedures as below and Sigma protocol is built automatically:
   1. Prover compute 1st message
   2. Prover compute 2nd message
   3. Verifier check
   4. Verifier query length
4. AND of any number of Sigma protocols [HL, Section 6.4]
5. OR of any two Sigma protocols [HL, Section 6.4]

**Zero knowledge**

1. Zero-knowledge for every Sigma-protocol using any commitment [HL, Section 6.5.1]
2. ZKPOK for every Sigma-protocol using any trapdoor commitment [HL, Section 6.5.2]

**Coin tossing**

1. Basic Blum single-coin tossing using any commitment scheme
   1. P1 commits to a single random bit using any commitment scheme
   2. P2 sends a random bit to P1
   3. P1 decommits
   4. Both parties output XOR of bits
2. [Lindell01] coin tossing, using Pedersen commitments and DLOG-ZK
   1. P1 commits to a random r using Pedersen
   2. P1 proves in ZKPOK that it knows the committed value (item 4 in sigma)
   3. P2 sends a random s
   4. P1 sends r (without decommitting)
   5. P1 proves in ZKPOK that r is the committed value (item 6 in sigma)
   6. Both parties output XOR of r and s
3. Semi-simulatable coin-tossing
   1. P1 sends a perfectly-hiding commitment to r (e.g. Pedersen or random-oracle)
   2. P2 sends a perfectly-binding commitment to s (e.g., Public-key commit or random-oracle)
   3. P1 opens
   4. P2 opens
   5. Both parties output XOR of r and s

**Oblivious transfer**

1. Naor-Pinkas (using any DH group) [HL, Section 7.2.1]
2. AIR (using any homomorphic encryption) [HL, Section 7.2.2]
3. HL-one-sided (using any DH group) [HL, Section 7.3]
4. HL-full simulation (using any DH group) [HL, Section 7.4]
5. PVW\_plain (using any DH group or N-residuosity) [HL, Section 7.5] and [PVW]

**Batch OT**

1. HL-full-sim [HL, Section 7.4.2]
2. PVW-batch [HL, Section 7.5]

**Oblivious polynomial evaluation**

**Oblivious pseudorandom function evaluation**

1. Based on OT [HL, Section 7.6]

**Authenticated communication channels**: use openSSL

**Private communication channels**: use openSSL

**Authenticated broadcast channel**

As in document by Meital

# Important Note

A crucial part of the design phase will be a pseudocode write-up of all the required primitives. This is to ensure that the translation from a protocol described in a paper to one that is to be implemented is correct.

## Bibliography

Katz-Lindell: Introduction to Modern Cryptography

HL: Hazay Lindell, Efficient Secure Two-Party Computation