6.897: Selected Topics in Cryptography

Lectures 11 and 12

Lecturer: Ran Canetti

Highlights of last week's lectures

- Formulated the ideal commitment functionality for a single instance, F_{com}.
- Showed that it's impossible to realize F_{com} in the plain model (even when given ideal authentication).
- Formulated the "CRS model" as the F_{crs}-hybrid model.
- Showed how to realize F_{com} in the F_{crs}-hybrid model.
- Showed how to do multiple commitments with the same CRS:
  - Formulated the multi-instance ideal commitment functionality, F_{mcom}.
  - Showed how to realize F_{mcom} given a single copy of F_{crs}.

This week:

- Show how to obtain UC ZK from UC commitments (this is “easy”, or “information-theoretic”)
- Show how to realize any multi-party functionality, for any number of faults, in the F_{crs}-hybrid model (using the [GMW87] paradigm).
- Mention how can be done in the plain model when there is honest majority (using elements from [BGW88]).

UC Zero-Knowledge from UC commitments

- Recall the ZK ideal functionality, F_{zk}, and the version with weak soundness, F_{wzk}.
- Recall the Blum Hamiltonicity protocol
- Show that, when cast in the F_{com}-hybrid model, a single iteration of the protocol realizes F_{wzk}.
  (This result is unconditional, no reductions or computational assumptions are necessary.)
- Show that can realize F_{zk} using k *parallel* copies of F_{wzk}.

The ZK functionality F_{zk} (for relation H(G,h)).

1. Receive (sid, P,V,G,h) from (sid,P).

   Then:

   1. Output (sid, P, V, G, H(G,h)) to (sid,V)
   2. Send (sid, P, V, G, H(G,h)) to S
   3. Halt.

The ZK functionality with weak soundness, F_{wzk} (for relation H(G,h)).

1. Receive (sid, P, V,G,h) from (sid,P).
   Then:
   1. If P is uncorrupted then set $v \leftarrow H(G,h)$.
   2. If P is corrupted then:
      - Choose $b \leftarrow_R \{0,1\}$ and send to S.
      - Obtain a bit b' and a cycle h' from S.
      - If $H(G,h')=1$ or $b'=b=1$ then set $v \leftarrow 1$. Else $v \leftarrow 0$.
   3. Output (sid, P, V, G,v) to (sid,V) and to S.
   4. Halt.

The Blum protocol in the F_{com}-hybrid model ("single iteration")

Input: sid,P,V, graph G, Hamiltonian cycle h in G.

- P → V: Choose a random permutation p on [1..n]. Let b_i be the i-th bit in p(G).p. Then, for each i send to F_{com}: (sid.i,P,V,"Commit",b_i) .
- V → P: When getting "receipt", send a random bit c.
- P → V:

  If c=0 then open all commitments

  (I.e., send F_{com}: (sid.i,"Open") for all I).

  If c=1 then open only commitments of edges in h.
- V accepts if all the commitment openings are received from F_{com} and in addition:
  - If c=0 then the opened graph and permutation match G
  - If c=1, then h is a Hamiltonian cycle.

Claim: The Blum protocol securely realizes F_{wzk}^{H} in the F_{com}–hybrid model

Proof sketch: Let A be an adversary that interacts with the protocol. Need to construct an ideal-process adversary S that fools all environments. There are four cases:

1. A controls the verifier (Zero-Knowledge):

S gets input z' from Z, and runs A on input z'. Next:

– If value from F_{zk} is (G,0) then hand (G,"reject") to A.
  If value from F_{zk} is (G,1) then simulate an interaction for V:
  - For all i, send (sid_i, "receipt") to A.
  - Obtain the challenge c from A.
  - If c=0 then send openings of a random permutation of G to A
  - If c=1 then send an opening of a random Hamiltonian tour to A.

The simulation is perfect…

2. A controls the prover (weak extraction):

S gets input z’ from Z, and runs A on input z’. Next:

I. Obtain from A all the “commit” messages to F_{com} and record the committed graph and permutation. Send (sid,P,V,G,h=0) to F_{wzk}.

II. Obtain the bit b from F_{wzk}.

If b=1 (i.e., F_{wzk} is going to allow cheating) then send the challenge c=0 to A.

If b=0 (I.e., no cheating allowed) then send c=1 to A.

III. Obtain A’s opening of the commitments in step 3 of the protocol.

If c=0, all openings are obtained and are consistent with G, then send b’=1 to F_{wzk} . If some openings are bad or inconsistent with G then send b’=0 (I.e., no cheating, and V should reject.)

If c=1 then obtain A’s openings of the commitments to the Hamiltonian cycle h’. If h’ is a Hamiltonian cycle then send h’ to F_{wzk}. Otherwise, send h’=0 to F_{wzk} .

Analysis of S: (A controls the prover):

The simulation is perfect. That is, the joint view of the simulated A together with Z is identical to their view in an execution in the F_{com} –hybrid model:

– V's challenge c is uniformly distributed.

– If c=0 then V's output is 1 iff A opened all commitments and the permutation is consistent with G.

– If c=1 then V's output is 1 iff A opened a real Hamiltonian cycle in G.

3. A controls neither party or both parties: Straightforward.

4. Adaptive corruptions: Trivial… (no party has any secret state).

■

From F_{wzk}^R to F_{zk}^R

A protocol for realizing F_{zk}^R in the F_{wzk}^R-hybrid model:

- P(x,w): Run k copies of F_{wzk}^R, *in parallel.* Send (x,w) to each copy.
- V: Run k copies of F_{wzk}^R, *in parallel.* Receive (x_i,b_i) from the i-th copy. Then:
  - If all x's are the same and all b's are the same then output (x,b).
  - Else output nothing.

Analysis of the protocol

Let A be an adversary that interacts with the protocol in the F_{wzk}^R-hybrid model. Need to construct an ideal-process adversary S that interacts with F_{zk}^R and fools all environments. There are four cases:

1. A controls the verifier: In this case, all A sees is the value (x,b) coming in k times, where (x,b) is the output value. This is easy to simulate: S obtains (x,b) from TP, gives it to A k times, and outputs whatever A outputs.
2. A controls the prover: Here, A should provide k inputs $x_1 \ldots x_k$ to the k copies of F_{wzk}^R, obtain k bits $b_1 \ldots b_k$ from these copies of F_{wzk}^R, and should give witnesses $w_1 \ldots w_k$ in return. S runs A, obtains $x_1 \ldots x_k$, gives it k random bits $b_1 \ldots b_k$, and obtains $w_1 \ldots w_k$. Then:
   - If all the x's are the same and all copies of F_{wzk}^R would accept, then find a w_i such that $R(x,w_i)=1$, and give (x,w_i) to F_{zk}^R. (If didn't find such w_i then fail. But this will happen only if $b_1 \ldots b_k$ are all 1, which occurs with probability 2^{-k}.)
   - Else give (x,w') to F_{zk}^R, where w' is an invalid witness.

Analysis of S:

- When the verifier is corrupted, the views of Z from both interactions are identically distributed.
- When the prover is corrupted, conditioned on the event that S does not fail, the views of Z from both interactions are identically distributed. Furthermore, S fails only if $b_1 \ldots b_k$ are all 1, and this occurs with probability 2^{-k}.

■

Note: The analysis is almost identical to the non-concurrent case, except that here the composition is done in parallel.

How to realize any two-party functionality

Based on [C-Lindell-Ostrovsky-Sahai02],
which is based on [GMW87].

Full version on eprint:
http://eprint.iacr.org/2002/140

How to realize any two-party functionality: The [GMW87] paradigm

1) Construct a protocol secure against *semi-honest* adversaries (i.e., even the corrupted parties follow the protocol specification).

2) Construct a general *compiler* that transforms protocols secure against semi-honest adversaries to “equivalent” protocols secure against Byzantine adversaries.

How to realize any two-party functionality: The [GMW87] paradigm

1) Construct a protocol secure against *semi-honest* adversaries (i.e., even the corrupted parties follow the protocol specification).

2) Construct a general *compiler* that transforms protocols secure against semi-honest adversaries to "equivalent" protocols secure against Byzantine adversaries.

(We'll first deal with two-party functionalities and then generalize to the multi-party case.)

The semi-honest, two-party case

- Few words about semi-honest adversaries.
- Present the ideal oblivious transfer functionality, F_{OT}.
- Show how to realize F_{OT} for semi-honest adversaries (in the plain model).
- Show how to realize "any functionality" in the F_{OT}-hybrid model.

The semi-honest adversarial model

- There are two “natural” variants:
  - The adversaries can change the inputs of the corrupted parties, but are otherwise passive
  - The environment talks directly with parties, adversaries only listen (cannot even change the inputs).
- The variants are incomparable…
- We’ll need the *first* variant for the compiler.
- The protocol we’ll present is secure according to both variants.

The (1-out-of-m) oblivious transfer functionality, F_{OT}^{m}.

1. Receive (sid, T, R, $v_1 \ldots v_m$) from (sid,T).
2. Receive (sid, R, T, i in {1..m}) from (sid,R).
3. Output (sid, v_i) to (sid,R).
4. Halt.

Realizing F_{OT}^2 (the [EGL85] protocol)

Let F be a family of trapdoor permutations and let B() be a hardcore predicate for F. Then:

Step 1: T (on input (v_0,v_1)), chooses f,f^{-1} from F, and sends f to R.

Step 2: R (on input i in {0,1}) chooses x_0,x_1, sets $y_i=f(x_i)$, $y_{1-i}=x_{1-i}$, and sends (y_0,y_1) to T.

Step 3: T computes $t_i=v_i+B(f^{-1}(y_i))$ and sends (t_0,t_1) to R.

Step 4: R outputs $v_i=t_i+B(x_i)$.

Theorem: The [EGL85] protocol realizes F_{OT}^2
For *semi-honest* adversaries with *static* corruptions.

Proof: For any A, construct an S that fools all Z…

S runs A. Then:

Corrupted sender: The information that A sees when observing T running the protocol is T's input (v_0,v_1), plus two values (y_0,y_1) received from R. S simulates this view, where (v_0,v_1) are taken from T's input in the ideal process and (y_0,y_1) are generated randomly.

Corrupted receiver: The information that A (observing R) sees is R's input i, the function f received from T, and the bits (t_0,t_1). Here S does:

- Obtains v_i from F_{OT}^2.
- Simulates for A a run of R on input i (taken from the ideal process). The simulated R receives a random f, generates (x_0,x_1), sends (y_0,y_1), and receives (t_0,t_1) where $t_i=v_i+B(x_i)$ and t_{1-I} is random.

Analysis of S:

- When the sender is corrupted, the simulation is perfect.
- When the receiver is corrupted, the validity of the simulation is reduced to the security of B and F:
  - Assume we have an environment that distinguishes between real and ideal executions, can construct a predictor that distinguishes between (f(x),B(x)) and (f(x),r) where x,r are random.

■

Remarks:

- Generalizes easily to n-out-of-m OT.
- To transfer k-bit values, invoke the protocol k times.
- For adaptive adversaries with erasures the same protocol works. Without erasures need to do something slightly different.

Evaluating general functionalities in the semi-honest, two-party case

Preliminary step:

Represent the ideal functionality F as a Boolean circuit:

- Assume "standard functionalities" (have "shell" and "core", where the "core" does not know who is corrupted.) We'll deal with the "core" only.
- Use " + " and " * " gates.
- Five types of input lines: Inputs of P_0, inputs of P_1, inputs of S, random inputs, local-state inputs.
- Four types of output lines: Outputs to P_0, P_1, outputs to S, local state for next activation.

The protocol in the F_{OT}-hybrid model

Step 1: Input sharing.

- When P_i is activated with new input, it notifies P_{1-i} and:
  - Shares each input bit b with P_{1-i}: sends $b_{1-i} \leftarrow_R \{0,1\}$ to P_{1-i} and keeps $b_i = b + b_{1-i}$.
  - For each random input line r, chooses $r_0, r_1 \leftarrow_R \{0,1\}$ and sends r_{1-I} to P_{1-i} .
  - In addition, P_i has its share s_i of each local state line s from the previous activation. (Initially, these shares are set to 0.)
  - P_i's shares of the adversary input lines are set to 0.
- When P_i is activated by notification from P_{1-i} it proceeds as above, except that it sets its inputs to be 0.

(At this point, the values of all input lines to the circuit are shared between the parties.)

The protocol in the F_{OT}-hybrid model

Step 2: Evaluating the circuit.

The parties evaluate the circuit gate by gate, so that the output value of each gate is shared between the parties: (Let a,b denote the input values of the gate, and let c denote the output value)

" + " gate: We have a+b=c. P_i has a_i and b_i, and computes $c_i=a_i+b_i$.

(Since $a_0+a_1=a$ and $b_0+b_1=b$, we have $c_0+c_1=c$.)

" * " gate: We have a*b=c. P_0 and P_1 use F_{OT}^4 as follows:

- P_0 chooses c_0 at random, and plays the sender with input:
  $(v_{00}= a_0b_0+c_0,\ v_{01}= a_0(1-b_0)+c_0\ ,\ v_{10}= (1-a_0)b_0+c_0\ ,\ v_{11}= (1-a_0)(1-b_0)+c_0)$
- P_1 plays the receiver with input (a_1,b_1), and sets the output to be c_1.

(Easy to verify that $c_0+c_1=(a_0+a_1)\ (b_0+b_1)$.)

The protocol in the F_{OT}-hybrid model

Step 3: Output generation

Once all the gates have been evaluated, each output value is shared between the parties. Then:

- P_{1-i} sends to P_i its share of the output lines assigned to P_i.
- P_i reconstructs its outputs and outputs them.
- P_i keeps its share of each local-state line (and will use it in the next activation).
- Outputs to the adversary are ignored.

Theorem:

Let F be a standard ideal functionality. Then the above protocol realizes F in the F_{OT}-hybrid model for semi-honest, adaptive adversaries.

Proof (very rough sketch):

For any A, construct S that fools all Z.

In fact, the simulation will be *unconditional* and *perfect*

(i.e., Z's views of the two interactions will be identical):

- The honest parties obtain the correct function values as in the ideal process.
- P_0 sees only random shares of input values, plus its outputs. This is easy to simulate.
- P_1 receives in addition also random shares of all intermediate values (from F_{OT}). This is also easy to simulate.
- Upon corruption, easy to generate local state.

Remarks:

- There is a protocol [Yao86] that works in constant number of rounds:
  - Can be proven for static adversaries (although havn't yet seen a complete proof)
  - Works also for adaptive adversaries with erasures.

- What about adaptive adversaries without erasures? Is there a general construction with constant number of rounds in this case?

[GMW87] Protocol Compilation

- Aim: force the malicious parties to follow the protocol specification.
- How?
  - Parties **commit** to inputs
  - Parties **commit** to *uniform* random tapes (use secure coin-tossing to ensure uniformity)
  - Run the original protocol Q, and in addition the parties use **zero-knowledge** protocols to prove that they follow the protocol. That is, each message of Q is followed be a ZK proof of the NP statement:

"There exist input x and random input r that are the legitimate openings of the commitments I sent above, and such that the message I just sent is a result of running the protocol on x,r, and the messages received so far".

Constructing a UC "[GMW87] compiler"

- Naive approach to solution:
  - Construct a GMW compiler given access to the ideal Commitment and ZK functionalities.
  - Compose with protocols that realize these functionalities.
  - Use the composition theorem to deduce security.

- Problem: If ideal commitment is used, there is no commitment string to prove statements on…

The “Commit&Prove” primitive

- Define a single primitive where parties can:
  - Commit to values
  - Prove “in ZK” statements regarding the committed values.

The Commit&Prove functionality, F_{cp} (for relation R)

1. Upon receiving (sid,C,V,“commit”,w) from (sid,C), add w to the list W of committed values, and output (sid,C, V, “receipt”) to (sid,V) and S.
2. Upon receiving (sid,C,V,“prove”,x) from (sid,C), send (sid,C,V,x,R(x,W)) to S. If R(x,W) then also output (sid,x) to (sid,V).

Note:

- V is assured that the value x it received in step 2 stands in the relation with the list W that C provided earlier
- C is assured that V learns nothing in addition to x and R(x,W).

Realizing F_{cp}^{R} in the F_{zk}-hybrid model

The protocol uses COM, a perfectly binding, non-interactive commitment scheme.

Protocol moves:

- To commit to w, (sid,C) computes a=COM(w,r), adds w to the list W, adds a to the list A, adds r to the list R, and sends (sid,C, V, "prove", a, (w,r)) to F_{zk}^{Rc} , where

  $$R_c=\{(a,(w,r)) : a=COM(w,r)\}.$$

- Upon receiving (sid, C, V, a,1) from F_{zk}^{Rc}, (sid,V) adds a to the list A, and outputs (sid,C,V,"receipt").

- To give x and prove R(x,W), (sid,C) sends (sid,C,V,"prove",(x,A),(W,R)) to F_{zk}^{Rp}, where

  $$R_p=\{((x,A),(W,R)) : W=w_1..w_n,\ A=a_1..a_n,\ R=r_1..r_n,\ R(x,W)\ \&\ a_i=COM(r_i;w_i) \text{ for all } i\}.$$

- Upon receiving (sid,C,V,(x,A),1) from F_{zk}^{Rp}, (sid,V) verifies that A agrees with its local list A, and if so outputs (sid,C,V,x).

Theorem:

The above protocol realizes F_{cp}^{R} in the F_{cp}-hybrid model for *non-adaptive* adversaries (assuming the security of COM).

Proof: For any A, construct an S that fools all Z…

S runs A. Then:

Corrupted committer:

- Commit phase: S obtains from A the message (sid,C, V, "prove", a, (w,r)) to F_{zk}^{Rc}. If Rc holds (I.e. a=COM(w,r)) then S inputs (sid,C,V,"commit",w) to F_{cp}.
- Prove phase: S obtains from A the message (sid,C,V,"prove",(x,A),(W,R)) to F_{zk}^{Rp}. If Rp holds then S inputs (sid,C,V,"prove",x) to F_{cp}.

Corrupted verifier:

- Commit phase: S obtains from F_{cp} a (sid,C,V,"receipt") message, and simulates for A the message (sid,C, V, a) from F_{zk}^{Rc}, where a=COM(0,r).
- Prove phase: S obtains from F_{cp} a (sid,C,V,"prove", x) message, and simulates for A the message (sid,C, V, (x,A)) from F_{zk}^{Rp}, where A is the list of simulated commitments generated so far.

Analysis of S:

Corrupted committer: Simulation is perfect.

Corrupted verifier: The only difference between the simulated and read executions is that in the simulation the commitment is to 0 rather than to the witness. Thus, if Z distinguishes then can construct an adversary that breaks the secrecy of the commitment.

■

Remarks:

- The proof fails in case of adaptive adversaries (even erasing will not help…)
- Can have an adaptively secure protocol by using "equivocable commitments".
- Can F_{cp} be realized unconditionally?

The compiler in the F_{cp}-hybrid model

Let $P=(P_1,P_2)$ be a protocol that assumes semi-honest adversaries. Construct the protocol $Q=C(P)$. The protocol uses two copies of F_{cp}, where in the i-th copy Q_i is the prover. Q_1 Proceeds as follows: (Q_2's code is analogous.)

1. Committing to Q_1's randomness (done once at the beginning):
   - Q_1 chooses random r_1 and sends (sid.1, Q_1,Q_2,"commit", r_1) to F_{cp} .
   - Q_1 receives r_2 from Q_2, and sets $r=r_1+r_2$.
2. Committing to Q_2's randomness (done once at the beginning):
   - Q_1 receives (sid.2,Q_2,Q_1,"receipt") from F_{cp} and sends a random value s_1 to Q_2.
3. Receiving the i-th new input, x:
   - Q_1 sends (sid.1, Q_1,Q_2,"commit", x) to F_{cp} .
   - Let M be the list of messages received so far. Q_1 runs the protocol P on input x, random input r, and messages M, and obtains either:
     - A local output value. In this case, output this value.
     - An outgoing message m. In this case, send (sid.1,Q_1,Q_2,"prove",m) to F_{cp} , where the relation is

$$R_P=\{((m,M,r_2),(x,r_1)) : m= P_1(x, r_1+r_2,M)\}$$

4. Receiving the i-th message, m:
   - Q_1 receives (sid.2, Q_2,Q_1,"prove", (m,M, s_1)) from F_{cp} . It verifies that s_1 is the value sent in Step 2, and that M is the set of messages sent to Q_2. If so, then run P_1 on incoming message m and continue as in Step 3.

Theorem:

Let P be a two-party protocol. Then the protocol Q=C(P), run with Byzantine adversaries, emulates protocol P, when run with semi-honest adversaries.

That is, for any Byzantine adversary A there exists a semi-honest adversary S such that for any Z we have:

$$\mathrm{Exec}_{P,S,Z} \sim \mathrm{Exec}^{F_{cp}}_{Q,A,Z}$$

Corollary:

If protocol P securely realizes F for semi-honest adversaries then Q=C(P) securely realizes F in the F_{cp}-hybrid model for Byzantine adversaries.

Proof: Will skip. But:

- Is pretty straightforward
- Is unconditional (and perfect simulation).
- Works even for adaptive adversaries
- Requires S to be able to change the inputs to parties.

Extension to the multiparty case: Challenges

- How to do the basic, semi-honest computation?
- Deal with asynchrony, no guaranteed message delivery.
- Deal with broadcast / Byzantine agreement
- How to use the existing primitives (OT, Com, ZK, C&P)?
- How to deal with variable number of parties?

Extension to the multiparty case: The semi-honest protocol
(fixed set of n parties)

Essentially the same protocol as for two parties, except:

- Each party shares its input among all parties: $x=x_1+\ldots+x_n$.
- Each random input, local state value is shared among all parties in the same way.
- Evaluating an addition gate: Done locally by each party as before. We have:

$$(a_1+\ldots+a_n)+(b_1+\ldots+b_n)=(a_1+b_1)+\ldots+(a_n+b_n)$$

- Evaluating a multiplication gate:
  - Each pair i<j of parties engage in evaluating the same OT, where they obtain shares c_i ,c_j such that $c_i+c_j=(a_i+a_j)(b_i+b_j)$.
  - Each party sums its shares of all the OT's. If n is even then also adds a_ib_j to the result.

  (Justification on the board…)

- Output stage: All parties send to P_i their shares of the output lines assigned to P_i.

Extension to the multiparty case: Byzantine adversaries

- Extend all functionalities (Comm, ZK, C&P) to the case of multiple verifiers (i.e., 1-to-many commitments, ZK, C&P).
- Realize using a broadcast channel (modeled as an ideal functionality, F_{bc}.)
- Can realize F_{bc} in an asynchronous network with any number of faults, via a simple "two-round echo" protocol.

Example: The 1:M commitment functionality, $F_{com}^{1:M}$

1. Upon receiving (sid,C,$V_1 \ldots V_n$,"commit",x) from (sid,C), do:
    1. Record x
    2. Output (sid,C, $V_1 \ldots V_n$, "receipt") to (sid,V_1)…(sid,V_n)
    3. Send (sid,C, $V_1 \ldots V_n$, "receipt") to S
2. Upon receiving (sid,"open") from (sid,C), do:
    1. Output (sid,x) to (sid,V_1)…(sid,V_n)
    2. Send (sid,x) to S
    3. Halt.

Honest majority: Can do without the CRS

- If we have honest majority then can realize $F_{com}^{1:M}$ in the plain model, using known VSS (Verifiable Secret Sharing) protocols, e.g., the ones in [BenOr,Goldwasser,Wigderson88].