6.897: Selected Topics in Cryptography

Lecturer: Ran Canetti

Lectures 3 and 4:

ZK as function evaluation and sequential composition of ZK

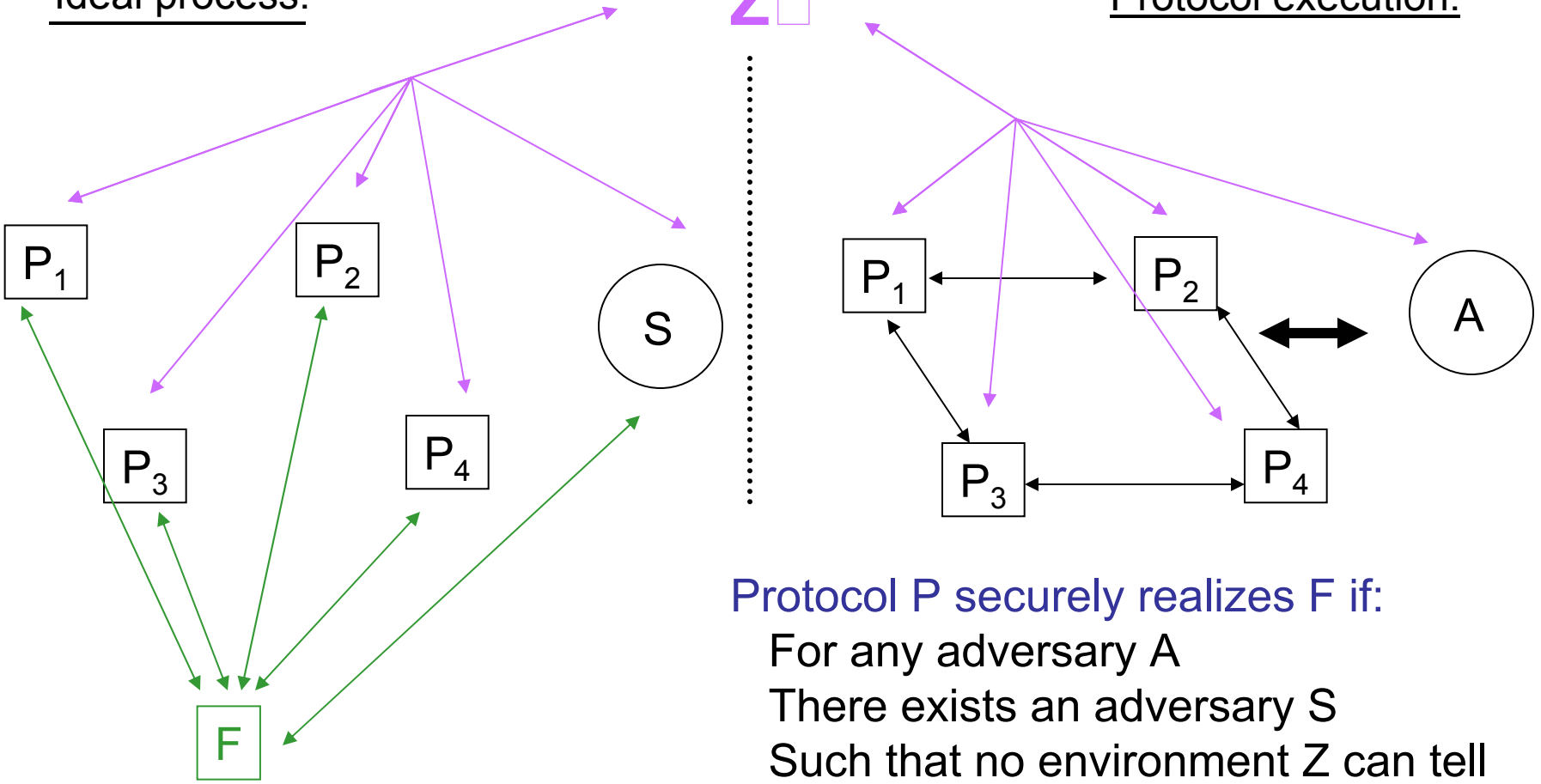
- Review the definition of ZK and PoK
- Give SFE-style definition of ZK and show equivalence to the standard one.
- The Blum protocol for Hamiltonicity:
  - Commitment schemes
  - The protocol
  - Analysis of a single instance: Weak soundness
  - Analysis of the iterated protocol using the composition theorem.

Review of the definition:

Z

Ideal process:

Protocol execution:



Protocol P securely realizes F if:

For any adversary A

There exists an adversary S

Such that no environment Z can tell whether it interacts with:

- A run of π with A
- An ideal run with F and S

Review: Zero-Knowledge

[Goldwasser-Micali-Rackoff 85]

Let R(x,w) be a polytime relation. A ZK protocol for R is a protocol for two parties P,V, where P has (x,w), V has no input, and the following holds:

- Completeness:

  $\text{Prob}[(P(x,w),V)=(x,R(x,w))]\sim 1$

  (if P,V are honest then V outputs x and the value of R(x,w))

- Soundness (I):

  For all P* and all z, $\text{Prob}[(P(z),V)=(x,1) \wedge (R(x,w)=0 \text{ for all } w)]\sim 0$

Differences from the standard definition:

1. In the standard def, V has x as input, whereas here V has no input and instead it outputs P's input x. (But it is easy to translate a protocol from one def to the other.)
2. The standard definition of ZK doesn't make any completeness requirement when R(x,w)=0. Here we require that V outputs (x,0). (But this is easy to achieve, by having P send (x,"reject") to V.)

Review: Zero-Knowledge

[Goldwasser-Micali-Rackoff 85]

- Zero-Knowledge:

For any verifier V^* there exists a machine S such that for all x,w,z, $S(x,R(x,w),z) \approx V^*_{P(x,w)}(z)$.

"Whatever V^* can gather from interacting with P, it could have computed by itself given R(x,w)."

(Distribution ensembles D,D' are computationally indistinguishable (written $D \approx D'$) if for any polytime distinguisher A, for all c,d, large enough k, and a with $|a| < k^d$,

$$\text{Prob}_{x \leftarrow Dk,a}[A(1^k,a,x)=1] - \text{Prob}_{x \leftarrow D'k,a}[A(1^k,a,x)=1] < 1/k^c)$$

Review: Proof of Knowledge

[Goldwasser-Micali-Rackoff, Goldreich-Oren, Tompa-Woll, Bellare-Goldreich, Halevi-Micali, Lindell, …]

Let R(x,w) be a polytime relation. A PoK protocol for R is a protocol for two parties P,V, where P has (x,w), and the following holds:

- Soundness (II), or "extractability":

For any prover P* there exists a "knowledge extractor" E, such that for any z, we have E(z) = (t,x*,w*) and

$$t,(x^*,R(x^*,w^*)) \approx AT(P^*(z),V)),$$

where AT is the "augmented transcript" of an execution of (P*(z),V)), namely the sequence of exchanged messages, plus the output of V.

"V is assured that, in each accepting interaction, it is possible to explicitly extract the witness from the prover's input. Furthermore, the extracted witness is the "appropriate witness" for this interaction." (This specific formulation is in the spirit of the "enhanced PoK" of [Lindell 03].)

ZK PoK as an SFE task

Let R(x,w) be a binary relation.

Consider the 2-party function:

$$F_{zk}^{R}((x,w), -, -) = (-, (x,R(x,w)), (x,R(x,w)))$$

Theorem: A two-party protocol securely realizes F_{zk}^{R} (with respect to non-adaptive adversaries) if and only if it is a ZK PoK for R.

Proof: Homework.

Example:
Blum's protocol for Graph Hamiltonicity
[Blum 8?]

Commitment schemes

Intuitive idea [Blum 82]:

Commitment is a two-party, two-step process:

- Commitment, where the committer C gives a value x to the receiver V "in an envelope"
- Decommitment, where C "opens" the envelope for V.

We want to guarantee:

- Secrecy: at the end of the commitment step V "has no knowledge" of what x is.
- Binding: Once the commitment step is done, there is a single value x fixed, such that C can successfully decommit only the value x.

Commitment schemes: A formalization

A basic formalization (commitments for one bit):

A tuple (C,D,V_c,V_d) of ITMs is a bit commitment scheme if:

- Completeness: For all b in {0,1},
  $\text{Prob}[(C(b),V_c)\rightarrow(s_c,s_v), (D(s_c),V_d(s_v))\rightarrow(-,b)] \sim 1$
- Secrecy: For all V*, $(C(0),V^*) \approx (C(1),V^*)$.
- Binding: For all C*,D*,
  $\text{Prob}[(C^*,V_c)\rightarrow(s_c,s_v),$
  $(D^*(s_c,0),V_d(s_v))\rightarrow(-,0),$
  $(D^*(s_c,1),V_d(s_v))\rightarrow(-,1)] \sim 0$

The basic Blum protocol

Common relation:
HC(G,H)=1 if H is a Hamiltonian cycle in graph G.

Common input: k-node graph G

Secret witness for P: Hamiltonian cycle H in G.

- P(G,H): If HC(G,H)=0 then send (G,"reject") to V. Else, choose a random permutation p on [1..k] and send (G, C(p(G)), C(p)) to V.
- V: If received (G,"reject") then output (G,0). Else, send a random bit b to P.
- P: If b=0 then decommit to all commitments of message 1. If b=1 then open only the commitments of the edges in H.
- V: Accept (output (G,1)) if all the commitment openings are accepted and, if b=1, then also if H is a Hamiltonian cycle. Else reject (output (G,0)).

Towards analysis…

Want to show that the protocol realizes F_{zk}^{HC}.
But clearly it does not: It has a soundness error of ½.

- Question: How to achieve full soundness, and in particular obtain a PoK?
- Basic idea: repeat many times, and have the verifier accept if all repetitions succeed. The hope is that ZK will be preserved, and soundness error will diminish.
- Questions:
  - How to repeat? (sequentially? In parallel?)
  - How to analyze?

We show that sequential repetition works.

Traditionally, this is proven directly. We'll use the composition theorem.

Plan:

- Define a "ZK with weak PoK functionality," F_{wzk}, and show that the basic protocol realizes it.
- Show how to realize F_{zk} in the F_{wzk}-hybrid model.
- Use the composition theorem to deduce security of the composed protocol.

The "ZK with weak PoK" functionality

- Idea:

  Relax F_{zk} to allow the adversary to break soundness w.p. up to ½. Let R(x,y) be a binary relation. Recall:

$$F_{zk}^{R}((x,w), -,-) = (- ,(x,R(x,w)),(x,R(x,w)))$$

- Define:

$F_{wzk}^{R}((x,w), -,c)$ = If R(x,w)=1 then output (-,(,x,1),(x,1).
Else, if c="no cheat" output (,(,x,0),(x,0)).
If c="cheat" output (,(,x,b),(x,b)) for $b \leftarrow_R \{0,1\}$.

Claim: The basic Blum protocol securely realizes F_{wzk}^{H}.

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) do:

- Choose random bit b'
- If b'=0 then hand A: C(p(G)), C(p) for random p.
- If b'=1 then hand A: C(Q), where Q is a random Hamilt. cycle.
- If A's response equals b' then open the commitments.
Else repeat, up to k times.

Analysis of S: Standard…
(via reduction to to the secrecy of the commitments)

2. A controls the prover (weak extraction):

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

I. Gather info:

- Obtain first message (G,C(G'),C(p)) from A.
- Give b=0 to A, obtain opening G'_0, p_0
- Rerun A (on same rand. inp.), give b=1, obtain G'_1, p_1.

II. Decide on on value to F and on output:

- If all decommitments succeed, then construct the Hamilt. Cycle. (if fail then some commitment was broken). Then give (G,H) to the TP (as the prover), and hand Z the output of A from either the first or the second run (chosen at random).
- If all decomm. for one value of b succeed, then:
  - give (G,-) to TP as the prover, and send "cheat" on direct line.
  - Get output (G,b') from TP. If b'=0 then output the ouptut of A from the run where decommitments failed. If b'=1 output the ouptut of A from the run where all decommitments succeed.
- If for both values of b some decommitments failed, give (G,-) to TP as the prover, send "no cheat" on direct line, and hand Z the output of A from either the first or the second run (chosen at random).

2. A controls the prover (weak extraction):

   Analysis of S:

   – If A answers both challenges correctly then Z expects V to always accept.
   – If A answers one challenge correctly then Z expects V to accept w.p. ½.
   – If A answers no challenge correctly then Z expects V to never accept.
   – Furthermore, Z expects to see the matching output of A.
   – S guarantees the same view for Z. (Only possiblity of failure: if some commitment decommits in the two runs of A. But this occurs with negligible prob., otherwise the commitment scheme is broken.)

3. A controls neither party:

   S obtains output from TP, generates a corresponding transcript of the protocol execution, gives it to A. Then, outputs whatever A outputs. (This works since the output of the verifier in a “benign” execution is correct.)

4. A controls both parties: S runs A.
   (Here there are no secrets that S does not know.)

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, sequentially. Send (x,w) to each copy.
- V: Run k copies of F_{wzk}^R, sequentially. 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 (or alternatively output a default value, say $(x_0,1)$ for some x_0 for which a w_0 is known.)

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 give to F_{wzk}^R k pairs (x_i,w_i), plus k "cheat"/"no-cheat" values. S runs A, obtains the k sets of input, runs the code of F_{wzk}^R k times, (it also tosses coins when F_{wzk}^R does), and obtains $(x_1,b_1)\ldots (x_k,b_k)$. Then:
   - If all the x's and all the b's are identical and b=1, 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.) If b=0 then give (x,w) to to F_{zk}^R, where w is an invalid witness.
   - Else S gives (x_0,w_0) (the default value) to F_{zk}^R.

   Finally, S outputs whatever A outputs.

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 in all k iterations it received bad witnesses, was asked to cheat, and chose b=1. But this occurs with probability 2^{-k}.

■

Note:

1. The protocol and analysis are purely combinatorial ("information theoretic"), no computational issues involved. All the "computational issues" are pushed to the realization F_{wzk}^{R} and the composition theorem.
2. The same analysis works for a number of other ZK protocols with weak extraction. No need to reprove sequential composition all these protocols; only needs to prove that the single-iteration protocol securely realizes F_{wzk}^{R}.

What about parallel composition (for error reduction)?

Protocol: Run k copies of Blum's protocol in parallel.
Accept if all copies accept.

Can't use the composition theorem… what about direct proof?

Intuitively, protocol should still be "secure". But proof fails.
Specifically, the "Zero-Knowledge simulator" no longer works:
Success probability in each attempt becomes 2^{-k} .

➔ It is unknown whether this protocol is ZK…

➔ There are known examples of ZK protocols that stop being ZK when two copies are running in parallel [Goldreich-Krawczyk, Feige-Shamir].

Parallel composition of Blum's protocol

- Solution 1 [Goldreich-Kahan]:

  Add verifier commitment to challenge.

  $V \rightarrow P$: $C(b_1),\ldots,C(b_k)$

  $P \rightarrow V$: $C(p_1(G)),C(p_1)\ldots C(p_k(G)),C(p_k)$

  $V \rightarrow P$: $D(b_1 \ldots b_k)$

  $P \rightarrow V$: $D(p_1(G)),D(p_1)\ldots D(p_k(G)),D(p_k)$

  V: accept if all copies accept.

Parallel composition of Blum's protocol

This solves the ZK problem. But now soundness fails:

Assume P,V use the same commitment C, and C is "malleable", say given C(b) can generate C(1-b) Then, P* can:

- Receive V's commitments $c_1 \ldots c_k$
- Generate it commitments $c'_1 \ldots c'_k$ so that:
  - If c_i is opened as 1 then c'_i opens to a Hamitonian cycle
  - If c_i is opened as 0 then c'_i opens to a permutation of G.

This problem can be solved by using special commitments. But the general use of commitments failed.

Parallel composition of Blum's protocol

- Solution 2 [Brassard-Crepeau-Yung]:

  Use "equivocable commitments" (given secret key, can open commitments both ways):

  V→P: public commitment key + "proof of knowledge" of secret key

  P→V: $C(p_1(G)),C(p_1)\ldots C(p_k(G)),C(p_k)$

  V→P: $b_1 \ldots b_k$

  P→V: $D(p_1(G)),D(p_1)\ldots D(p_k(G)),D(p_k)$

  V: accept if all copies accept.

- In the simulation, "extract" the key, and open the commitments both ways.

→ Not a standard use of commitments (need an additional "weird-looking property"…)

So far, we've encountered two drawbacks of the current notions of ZK, commitment:

- ZK Not closed under parallel composition
- Use of commitments within ZK protocols is "non-intuitive" and "non-modular"