6.897: Selected Topics in Cryptography
Lectures 9 and 10

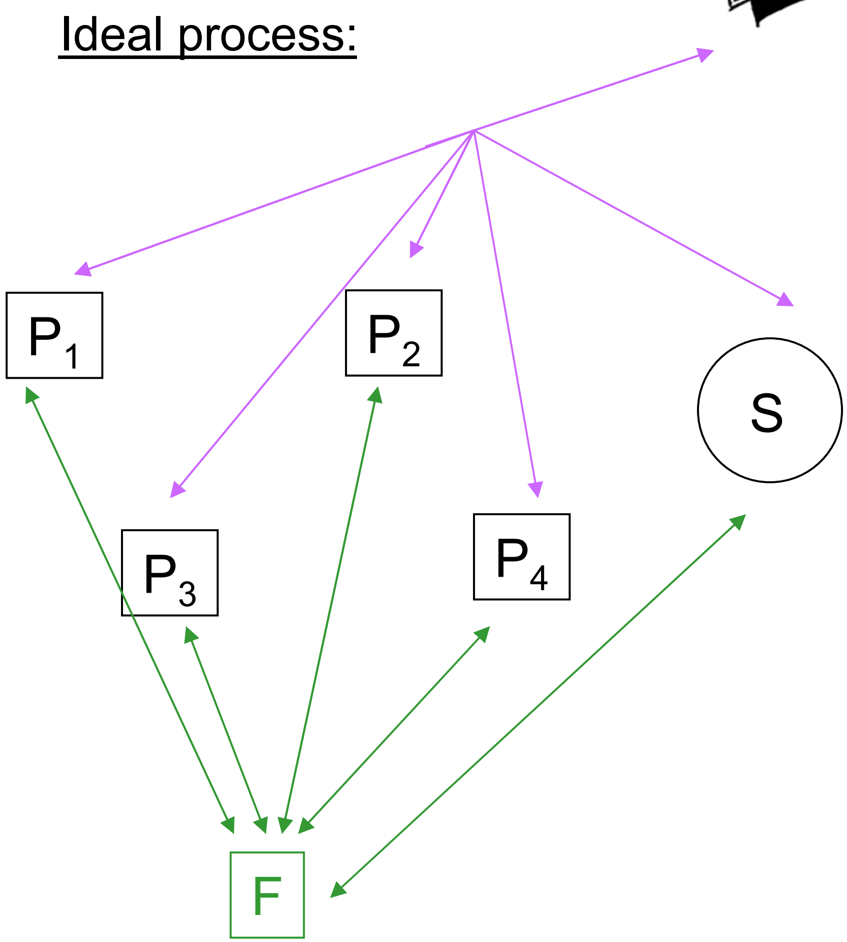
Lecturer: Ran Canetti

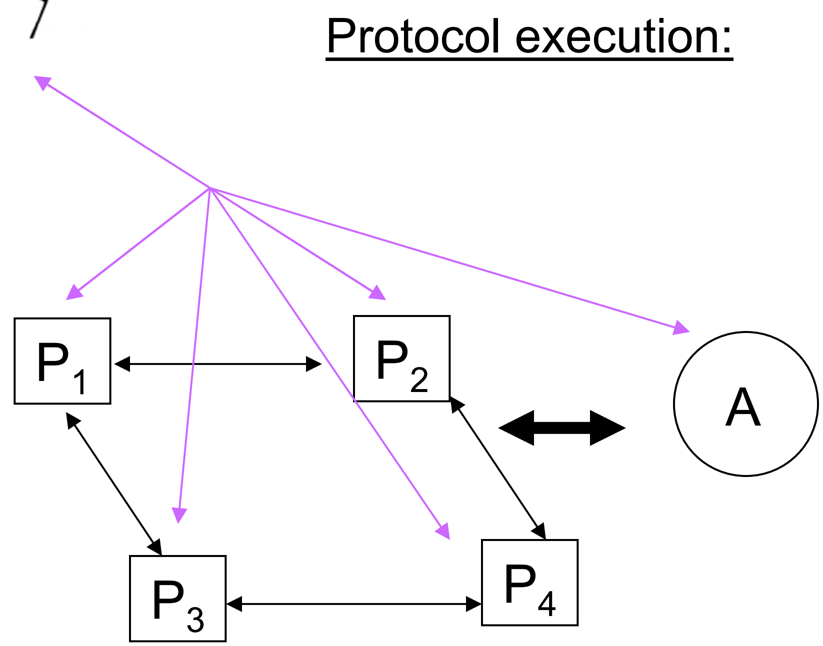
Highlights of past lectures

Presented two frameworks for analyzing protocols:

- A basic framework:
  - Only function evaluation
  - Synchronous
  - Non-adaptive corruptions
  - Modular composition (only non-concurrent)
- A stronger framework (UC):
  - General reactive tasks
  - Asynchronous (can express different types of synchrony)
  - Adaptive corruptions
  - Concurrent modular composition (universal composition)

Review of the definition:





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

Lectures 9 and 10
UC Commitment and Zero-Knowledge

- Quick review of known feasibility results in the UC framework.
- UC commitments: The basic functionality, F_{com}.
- Impossiblity of realizing F_{com} in the plain model.
- Realizing F_{com} in the common reference string model.
- Multiple commitments with a single string:
  - Functionality F_{mcom}.
  - Realizing F_{mcom}.
- From UC commitments to UC ZK:

  Realizing F_{zk} in the F_{com}-hybrid model.

Questions:

- How to write ideal functionalities that adequately capture known/new tasks?
- Are known protocols UC-secure?

  (Do these protocols realize the ideal functionalities associated with the corresponding tasks?)
- How to design UC-secure protocols?

Existence results: Honest majority

Multiparty protocols with honest majority:

Thm: Can realize *any functionality* [C. 01].

(e.g. use the protocols of [BenOr-Goldwasser-Wigderson88, Rabin-BenOr89,Canetti-Feige-Goldreich-Naor96]).

Two-party functionalities

- Known protocols do not work.
  ("black-box simulation with rewinding" cannot be used).
- Many interesting functionalities (commitment, ZK, coin tossing, etc.) cannot be realized in plain model.
- In the "common random string model" can do:
  - UC Commitment
    [Canetti-Fischlin01,Canetti-Lindell-Ostrovsky-Sahai02,Damgard-Nielsen02, Damgard-Groth03,Hofheinz-QuedeMueler04].
  - UC Zero-Knowledge [CF01, DeSantis et.al. 01]
  - Any two-party functionality [CLOS02,Cramer-Damgard-Nielsen03]

    (Generalizes to any *multiparty* functionality with any number of faults.)

UC Encryption and signature

- Can write a "digital signature functionality" F_{sig}. Realizing F_{sig} is equivalent to "security against chosen message attacks" as in [Goldwasser-Micali-Rivest88].
  - Using F_{sig}, can realize "ideal certification authorities" and "ideally authenticated communication".

- Can write a "public key encryption functionality", F_{pke}. Realizing F_{pke} *w.r.t. non-adaptive adversaries* is equivalent to "security against chosen ciphertext attacks (CCA)" as in [Rackoff-Simon91,Dolev-Dwork-Naor91,…].
  - Can formulate a relaxed variant of F_{pke}, that still captures most of the current applications of CCA security.
  - What about realizing F_{pke} w.r.t. adaptive adversaries?
    - As is, it's impossible.
    - Can relax F_{pke} a bit so that it becomes possible (but still very complicated) [Canetti-Halevi-Katz04]. How to do it simply?

UC key-exchange and secure channels

- Can write ideal functionalities that capture Key-Exchange and Secure-Channels.
- Can show that natural and practical protocols are secure: ISO 9798-3, IKEv1, IKEv2, SSL/TLS,…
- What about password-based key exchange?
- What about modeling symmetric encryption and message authentication as ideal functionalities?

UC commitments

The commitment functionality, F_{com}

1. Upon receiving (sid,C,V,“commit”,x) from (sid,C), do:
   1. Record x
   2. Output (sid,C,V, “receipt”) to (sid,V)
   3. Send (sid,C,V, “receipt”) to S
2. Upon receiving (sid,“open”) from (sid,C), do:
   1. Output (sid,x) to (sid,V)
   2. Send (sid,x) to S
   3. Halt.

Note: Each copy of F_{com} is used for a single commitment/decommitment Only. Multiple commitments require multiple copies of F_{com}.

Impossibility of realizing F_{com} in the plain model

F_{com} can be realized:

- By a "trivial" protocol that never generates any output.
  (The simulator never lets F_{com} to send output to any party.)
- By a protocol that uses third parties as "helpers".

→ A protocol is:

- Terminating, if when run between two honest parties, some output is generated by at least one party.
- Bilateral, if only two parties participate in it.

Theorem: There exist no terminating, bilateral protocols that securely realize F_{com} in the plain real-life model. (Theorem holds even in the F_{auth}-hybrid model.)

Proof Idea:

Let P be a protocol that realizes F_{com} in the plain model, and let S be an ideal-process adversary for P, for the case that the commiter is corrupted.

Recall that S has to explicitly give the committed bit to F_{com} before the opening phase begins. This means that S must be able to somehow “extract” the committed value b from the corrupted committer.

However, in the UC framework S has no advantage over a real-life verifier. Thus, a corrupted verifier can essentialy run S and extract the committed bit b from an honest committer, before the opening phase begins, in contradiction to the secrecy of the commitment.

More precisely, we proceed in two steps:

(I) Consider the following environment Z_C and real-life adversary A_C that controls the committer C:

- A_C is the dummy adversary: It reports to Z_C any message received from the verifier V, and sends to V any message provided by Z_C.
- Z_C chooses a random bit b, and runs the code of the honest C by instructing A_C to deliver all the messages sent by C.

  Once V outputs "receipt", Z_C runs the opening protocol of C with V, and outputs 1 if the output bit b' generated by V is equal to b.

From the security of P there exists an ideal-process adversary S_C such that $IDEAL^{Fcom}_{Sc,,Zc} \sim EXEC_{P,Ac,Zc}$. But:

- In the real-life mode, b', the output of V, is almost always the same as the bit b that secretly Z chose.
- Consequently, also in the ideal process, b'=b almost always.
- Thus, the bit b'' that S provides F_{com} at the commitment phase is almost always equal to b.

(II) Consider the following environment Z_V and real-life adversary A_V that controls the verifier V:

- Z_V chooses a random bit b, gives b as input to the honest commiter, and outputs 1 if the adversary output a bit b'=b.
- A_V runs S_C. Any message received from C is given to S_C, and any message generated by S_C is given to C. When S_C outputs a bit b' to be given to F_{com}, A_V outputs b' and halts.

Notice that the view of S_C when run by A_V is identical to its view when interacting with Z_C in the ideal process for F_{com}. Consequently, from part (I) we have that in the run of Z_V and A_V almost always b'=b.

However, when Z_V interacts with *any* simulator S in the ideal process for F_{com}, the view of S is independent of b. Thus Z_V outputs 1 w.p. at most ½.

This contradicts the assumption that P securely realizes F_{com}. $\blacksquare$

The common reference string functionality

Functionality F_{crs}

(with prescribed distribution D)

1. Choose a value r from distribution D, and send r to the adversary.
2. Upon receiving ("CRS",sid) from party P, send r to P.

Note: The F_{crs}-hybrid model is essentially the "common reference string model", as usually defined in the literature (cf., Blum-Feldman-Micali89).
In particular: An adversary in the F_{crs}-hybrid model expects to get the value of the CRS from the ideal functionality. Thus, in a simulated interaction, the simulator can choose the CRS by itself (and in particular it can know trapdoor information related to the CRS).

Theorem: If trapdoor permutation pairs exist then there exist terminating, bilateral protocols that realize F_{com} in the (F_{auth},F_{crs})-hybrid model.

Remarks:

- Here we'll only show the [CF01] construction, that is based on claw-free pairs of trapdoor permutations.
- [DG03] showed that UC commitments imply key exchange, so no black-box constructions from OWPs exist.
- More efficient constructions based on Paillier's assumption exist [DN02, DG03, CS03].

Realizing F_{com} in the F_{crs}-hybrid model

- Roughly speaking, we need to make sure that the ideal model adversary for F_{com} can:
  - Extract the committed value from a corrupted committer.
  - Generate commitments that can be opened in multiple ways.
  - Explain internal state of committer and verifier upon corruption (for adaptive security).

First attempt

- To obtain equivocability:
  - Let $f=\{f_0, f_1, f_0^{-1}, f_1^{-1}\}$ be a claw-free pair of trapdoor permutations. That is:
    - f_0, f_1 are over the same domain.
    - Given f_i and x it is easy to compute $f_i(x)$.
    - Given f_i^{-1} and x it is easy to compute $f_i^{-1}(x)$.
    - Given only f_0, f_1, it is hard to find x_0, x_1 such that $f_0(x_0)=f_1(x_1)$.
  - Commitment Scheme:
    - CRS: f_0,f_1
    - To commit to bit b, choose random x in the domain of f and send $f_b(x)$. To open, send b,x.
  - Simulator chooses the CRS so that it knows the trapdoors f_0^{-1},f_1^{-1}. Now can equivocate: find x_0,x_1 s.t. $f_0(x_0)=f_1(x_1)=y$, send y.
- But: Not extractable…

Second attempt

- To add extractability:
  - Let (G,E,D) be a semantically secure encryption scheme.
  - Commitment Scheme:
    - Let G(k)=(e,d). CRS: f_0,f_1, e.
    - To commit to a bit b, choose random x,r, and send $f_b(x)$,$E_e(r,x)$. To open, send b,x,r.
  - Simulator knows choose the CRS such that it knows the decryption key d. So it can decrypt and extract b.

- But: lost equivocability…

Third attempt

- To restore equivocability:
  - Scheme:
    - CRS: f_0,f_1, e
    - To commit to b:
      - choose random x,r_0,r_1
      - send $f_b(x),E_e(r_b,x),E_e(r_{1-b},0)$
    - To open, send b,x,r_b. (*Don't* send r_{1-b}.)
  - To extract, simulator decrypts both encryptions and finds x.
  - To equivocate, simulator chooses x_0,x_1,r_0,r_1, such that $f_0(x_0)=f_1(x_1)=y$ and sends $y,E_e(r_0,x_0),E_e(r_1,x_1)$.

The protocol (UCC) for static adversaries

- On input (sid,C,V,"commit",b) C does:
  - Choose random x,r_0,r_1. Obtain f_0,f_1, e from F_{crs}.
  - Compute $y= f_b(x)$, $c_b=E_e(r_b,x)$, $c_{1-b}=E_e(r_{1-b},0)$, and send (sid,C,V,y,c_0,c_1) to V.
- When receiving (sid,C,V,y,c_0,c_1) from C, V outputs (sid,C,"receipt",C).
- On input (sid,"open"), C does:
  - Send b,x,r_b to V.
- Having received b,x,r, V verifies that $F_b(x)=y$ and $c_b=E_e(r,x)$. If verification succeeds then output ("Open",sid,cid,C,b). Else output nothing.

Proof of security (static case)

Let A be an adversary that interacts with parties running protocol UCC in the F_{crs}-hybrid model.

We construct a simulator S in the ideal process for F_{com} and show that for any environment Z,

$$IDEAL^{F_{com}}_{S,Z} \sim EXEC_{ucc,A,Z}$$

Simulator S:

- Choose a c.f.p. (f_0, f_1, f_0^{-1}, f_1^{-1}) and keys (e,d) for the enc. Scheme.
- Run a simulated copy of A and give it the CRS (f_0, f_1, e).
- All messages between A and Z are relayed unchanged.
- If the committer C is uncorrupted:
  - If S is notified by F_{com} that C wishes to commit to party V then simulate for A a commitment from C to V: Choose y, compute $x_0=f_0^{-1}(y)$,$x_1= f_1^{-1}(y)$, $c_0=E_e(r_0,x_0)$, $c_1=E_e(r_1,x_1)$, and send (y, c_0, c_1) from C to V. When A delivers this message to V, send "ok" to F_{com}.
  - If S is notified by F_{com} that C opened the commitment to value b, then S simulates for A the opening message (b, x_b, r_b) from C to V.
- If C is corrupted:
  - If a corrupted C sends a commitment (y, c_0, c_1) to V, then S decrypts c_0 and c_1:
    - If c_0 decrypts to x_0 where $x_0=f_0^{-1}(y)$, then send (sid,C,V,"commit",0) to F_{com}.
    - If c_1 decrypts to x_1 where $x_1=f_1^{-1}(y)$, then send (sid,C,V,"commit",1) to F_{com}.
  - If C sends a valid opening message (b',x,r) (I.e., $x=f_{b'}^{-1}(y)$ and $c_{b'}=E_e(r,x)$), then S checks whether b' equals the bit sent to F_{com}. If yes, then S sends (sid, "Open") to F_{com}. Otherwise, S aborts the simulation.

Analysis of S:

Let Z be an environment. define first the following hybrid interaction HYB:

Interaction HYB is identical to $IDEAL^{Fcom}_{S,Z,}$ except that when S generates commitments by uncorrupted parties, it "magically learns" the real bit b, and then uses real (not fake) commitments. That is, the commitment is (y, c_0, c_1) where $c_{1-b}=E_e(r_{1-b},0)$.

We proceed in two steps:

1. Show that $EXEC_{ucc,A,Z} \sim HYB$.
   This is done by reduction to the security of the claw-free pair.
2. Show that $HYB \sim IDEAL^{Fcom}_{S,Z}$.
   This is done by reduction to the semantic security of the encryption scheme.

Step 1: Show that $EXEC_{ucc,A,Z}$ ~ HYB:

- Note that the interactions $EXEC_{ucc,A,Z}$ and HYB are identical, as long as the adversary does not abort in an opening of a commitment made by a corrupted party.
- We show that if S aborts with probability p then we can find claws in (f_0, f_1) With probability p. That is, construct the following adv. D:
  - Given (f_0, f_1), D simulates an interaction between Z and S (running A) when the c.f.p. in the CRS is (f_0, f_1). D plays the role of S for Z and A. Since D sees all the messages sent by Z, it knows the bits committed to be the uncorrupted parties, and can simulate the interaction perfectly.

    Furthermore, whenever S aborts then D finds a claw in (f_0, f_1): S aborts if A provides a valid commitment to a bit b and then a valid opening to 1-b. But in this case A generated a claw!

Step 2: Show that HYB ~ $IDEAL^{Fcom}_{S,Z}$:

Recall that the difference between HYB and $IDEAL^{Fcom}_{S,Z}$ is that in HYB the commitments generated by S are real, whereas in $IDEAL^{Fmcom}_{S,Z}$ these commitments are fake.

Assume an env. Z and adv. A that distinguish between the two interactions. Construct an adversary B that breaks the semantic security of (E,D):

Given encryption key e, B simulates an interaction between Z and S (running A) when the encryption key in the CRS is e. B plays the role of S for Z and A. Furthermore, When S needs to generates a commitment (y, c_0, c_1), B does:

- C_b is generated honestly as $c_b=E_e(r_b,x_b)$. (Recall, B knows b.)
- B asks its encryption oracle to encrypt one out of $(0, x_{1-b})$ and sets the answer C* to be c_{1-b}.

Analysis of B:

- If C*=E(0) then the simulated Z sees an HYB interaction.
- If C*=E(x_{1-b}) then the simulated Z sees an $IDEAL^{Fcom}_{S,Z}$ interaction.

Since Z distinguishes between the two, B breaks the semantic security of the encryption scheme. ∎

Dealing with adaptive adversaries

Recall the protocol (UCC) for static adversaries

- On input (sid,C,V,“commit”,b) C does:
  - Choose random x, r_0, r_1. Obtain f_0, f_1, e from F_{crs}.
  - Compute $y = f_b(x)$, $c_b = E_e(r_b, x)$, $c_{1-b} = E_e(r_{1-b}, 0)$, and send (sid, C, V, y, c_0, c_1) to V.
- When receiving (sid, C, V, y, c_0, c_1) from C, V outputs (sid,C,“receipt”,C).
- On input (sid,“open”), C does:
  - Send b, x, r_b to V.
- Having received b,x,r, verifies that $F_b(x) = y$ and $c_b = E_e(r, x)$. If verification succeeds then output (“Open”,sid,cid,C,b). Else output nothing.

Problem: When the committer is corrupted, it needs to present the randomness r_{1-b}. Now S is stuck…

Solutions:

- Erase r_{1-b} immediately after use inside the encryption.
- If do not trust erasures: Use an encryption where ciphertexts are "pseudorandom". Then the commitment protocol changes to:
  - Choose random x, r_0, r_1. Obtain f_0, f_1, e from F_{crs}.
  - Let $y = f_b(x)$, $c_b = E_e(r_b, x)$, $c_{1-b} = r_{1-b}$, and send (sid, C, V, y, c_0, c_1) to V.

  Simulation changes accordingly.

Note: Secure encryption with pseudorandom ciphertexts exists given any trapdoor permutation: Use the Goldreich-Levin HardCore bit.

How to re-use the CRS?

Functionality F_{com} handles only a single commitment. Thus, to obtain multiple commitments one needs multiple copies of F_{com} . When replacing each copy of F_{com} with a protocol P that realizes it in the F_{crs}-hybrid model, one obtains multiple copies of P, which in turn use multiple independent copies of F_{crs}…

- Can we realize multiple copies of F_{com} using a single copy of F_{crs}?
- How to formalize that?

The multi-instance commitment functionality, F_{mcom}

1. Upon receiving (sid,cid,C,V,“commit”,x) from (sid,C), do:
   1. Record (cid,x)
   2. Output (sid,cid,C,V, “receipt”) to (sid,V)
   3. Send (sid,cid,C,V, “receipt”) to S
2. Upon receiving (sid,cid“open”) from (sid,C), do:
   1. Output (sid,cid,x) to (sid,V)
   2. Send (sid,cid,x) to S

How to realize F_{mcom}?

- Trivial solution: Run multiple copies of protocol ucc, where each copy uses its own copy of F_{crs}…
- But, can we do it with a single copy of F_{crs}?
- Does protocol ucc do the job?

Attempt 1: Run as is.

Bad: Adversary can copy commitments.

Attempt 2: Include the committer's id inside the encryption. I.e., in the commitment phase compute $c_b=E_e(r_b,C.x)$, $c_{1-b}=E_e(r_{1-b},C.0)$.

Bad: Adversary can change the encrypted id inside c_0,c_1.

Attempt 3: Use CCA2 ("non-malleable") encryption.

Works…

The protocol (UCMC) for static adversaries

- On input ("commit",V,b,sid,cid) C does:
  - Choose random x, r_0, r_1. Obtain f_0, f_1, e from F_{crs}.
    (Now e is the encryption key of a CCA2-secure encryption scheme.)
  - Compute $y= f_b(x)$, $c_b=E_e(r_b,C.x)$, $c_{1-b}=E_e(r_{1-b},C.0)$, and send (sid,cid,C,V,y,c_0,c_1) to V.
- When receiving (sid,cid,C,V,y,c_0,c_1) from C, V outputs ("receipt",C,sid,cid).

- On input ("open",sid,cid), C does:
  - Send b,x,r_b to V.
- Having received b,x,r_b, V verifies that $F_b(x)=y$ and $c_b=E_e(r_b,C.x)$, and that cid never appeared before in a commitment of C.

If verification succeeds then output ("Open",sid,cid,C,b).

Else output nothing.

Proof of security (static case)

- The simulator S is identical to that of UCC, except that here it handles multiple commitments and decommitments.
- Analysis of S:
  - Define the same hybrid interaction HYB.
  - The proof that $EXEC_{ucc,A,Z} \sim HYB$ remains essentally the same, except that here there are many commitments and decommitments.
  - The proof that $HYB \sim IDEAL^{Fmcom}_{S,Z}$ is similar in structure to the proof for the single commitment case, except that here the reduction is to the CCA security of the encryption:

Simulator S:

- Choose a c.f.p. (f_0, f_1, f_0^{-1}, f_1^{-1}) and keys (e,d) for the enc. Scheme.
- Run A and give it the CRS (f_0, f_1, e).
- All messages between A and Z are relayed unchanged.
- Commitments by uncorrupted parties:
  - If S is notified by F_{mcom} that an uncorrupted C wishes to commit to party V with a given cid, then simulate for A a commitment from C to V: Choose y, compute $x_0=f_0^{-1}(y)$,$x_1= f_1^{-1}(y)$, y, $c_0=E_e(r_0,C.x_0)$, $c_1=E_e(r_1,C.x_1)$, and send (y, c_0, c_1) from C to V. When A delivers this message to V, send "ok" to F_{mcom}.
  - If S is notified by F_{mcom} that C opened the commitment cid to value b, then it simulates for A an opening message (b, x_b, r_b) from C to V.
- Commitments by corrupted parties:
  - If A sends a commitment (cid, y, c_0, c_1) in the name of a corrupted committer C to some V, then S decrypts c_0. If c_0 decrypts to C.x_0 where $x_0=f_0^{-1}(y)$, then let b=0. Else b=1. Then, send ("commit",C,V,b,sid,cid) to F_{mcom}.
  - If A sends a valid opening message (b',x,r) for some cid (I.e., $x=f_{b'}^{-1}(y)$, $c_{b'}=E_e(r,C.x)$), and b'=b, then S sends ("Open",sid,cid) to F_{mcom}.
    If b' != b, then S aborts the simulation

Analysis of S:

Let Z be an environment. define first the following hybrid interaction HYB:

Interaction HYB is identical to $IDEAL^{Fmcom}_{S,Z}$, except that when S generates commitments by uncorrupted parties, it "magically learns" the real bit b, and then uses real (not fake) commitments. That is, the commitment is (y, c_0, c_1) where $c_{1-b}=E_e(r_{1-b},C.0)$.

We proceed in two steps:

1. Show that $EXEC_{ucc,A,Z} \sim HYB$.
   This is done by reduction to the security of the claw-free pair.
2. Show that $HYB \sim IDEAL^{Fmcom}_{S,Z}$.
   This is done by reduction to the security of the encryption scheme.

Step 1: Show that $EXEC_{ucc,A,Z}$ ~ HYB:

- Note that the interactions $EXEC_{ucc,A,Z}$ and HYB are identical, as long as the adversary does not abort in an opening of a commitment made by a corrupted party.
- We show that if S aborts with probability p then we can find claws in (f_0, f_1) With probability p. That is, construct the following adv. D:
  - Given (f_0, f_1), D simulates an interaction between Z and S (running A) when the c.f.p. in the CRS is (f_0, f_1). D plays the role of S for Z and A. Since D sees all the messages sent by Z, it knows the bits committed to be the uncorrupted parties, and can simulate the interaction perfectly.

    Furthermore, whenever S aborts then D finds a claw in (f_0, f_1): S aborts if A provides a valid commitment to a bit b and then a valid opening to 1-b. But in this case A generated a claw!

Step 2: Show that HYB ~ $IDEAL^{Fmcom}_{S,Z}$:

Recall that the difference between HYB and $IDEAL^{Fmcom}_{S,Z}$ is that in HYB the commitments generated by S are real, whereas in $IDEAL^{Fmcom}_{S,Z}$ these commitments are fake.

Assume a env. Z that distinguishes between the two interactions. Construct a CCA-adversary B that breaks the security of (E,D). (In fact, B will interact in a Left-or-Right CCA interaction):

Given encryption key e, B simulates an interaction between Z and S (running A) when the encryption key the CRS is e. B plays the role of S for Z and A. Furthermore:

- When S needs to generates a commitment (y, c_0, c_1), B does:
  - C_b is generated honestly as $c_b=E_e(r_b,C.x_b)$. (Recall, B knows b.)
  - B asks its encryption oracle to encrypt one out of (0, $C.x_{1-b}$) and sets the answer to be c_{1-b}.
- When A sends a commitment (y, c_0, c_1), B does:
  - If either c_0 or c_1 are test ciphertexts then they can be safely ignored, since they contain an ID of an uncorrupted party. Else, B asks its decryption oracle to decrypt, and continues running S.

Note:

- If B's oracle is a "Left" oracle (ie, all the test ciphertexts are encryptions of ID.0) then the simulated Z sees an HYB interaction.
- If B's oracle is a "Right" oracle (ie, all the test ciphertexts are encryptions of ID. x_{1-b}) then the simulated Z sees an $\text{IDEAL}^{Fmcom}_{S,Z}$ interaction.

Since Z distinguished between the two, B breaks the LR-CCA security of the encryption scheme.

$\blacksquare$

Dealing with adaptive corruptions

Use the same trick as in the single-commitment case.

Question: How to obtain CCA-secure encryption with p.r. ciphertexts?

- Cramer-Shoup…
- Use double encryption: E(x)=E'(E"(x)), where:
  - E' is CPA-secure with p.r. ciphertext (e.g., standard encryption based on hard-core bits of tradoor permutations).
  - E" is CCA-secure.

Note: E is not CCA-secure, but is good enough…

UC Zero-Knowledge from UC commitments

- Recall the ZKPoK 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 ZKPoK 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 weak ZKPoK functionality F_{wzk} (for relation H(G,h)).

1. Receive (sid, P, V,G,h) from (sid,P).
   Then:
   1. 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, and replace $h \leftarrow h'$.
   2. 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 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.
  - If 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. If the bit b obtained from F_{wzk} is 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 in this run) 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 c=0 and some openings are bad or inconsistent with G then send b'=0 (I.e., no cheating, and V should not accept.)

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} .

2. A controls the prover (weak extraction):

Analysis of S:

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.

■

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, and this occurs with probability 2^{-k}.)
   - Else give (x,w') to 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 in parallel.