Today's topics:

- UC ZK from UC commitments (this is information theoretic and unconditional; no crypto needed)
- MPC, under any number of faults (using the paradigm of [GMW87])
- MPC in the plain model with an honest majority (using elements of [BOGW88] and [RBO89])

1 UC Zero Knowledge from UC Commitments

To implement F_{zk}, we will use the following ingredients: the "weak soundness" version of ZK, F_{wzk}; the Blum Hamiltonicity protocol; a proof that one instance of the protocol realizes F_{wzk} in the F_{com}-hybrid model; and a realization of F_{zk} using k *parallel* copies of F_{wzk}.

Let's recall the F_{wzk}^{H} functionality (i.e., "zero knowledge with weak soundness/extraction"). For simplicity, we work directly with the Hamiltonicity relation H, where $H(G,h)=1$ iff h is a Hamiltonian tour of a graph G. The intuition behind F_{wzk} is that, with probability 1/2, it allows a corrupted prover (and *only* a corrupted prover) to "cheat" if it so desires. When the prover cheats, F_{wzk} instructs the verifier to accept even if the prover did not supply a valid witness. The functionality is formally defined as follows:

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

   Finally, output $(\mathsf{sid}, P, V, G, v)$ to (sid, V) and to S, and halt.

[the Blum protocol?]

Claim 1 *The Blum protocol security realizes F_{wzk}^{H} in the F_{com}-hybrid model.*

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

1. If A controls the verifier (zero-knowledge): S gets input z' from Z, and runs A on z'. If the value from F_{zk} is $(G,0)$, then send (G, reject) to A. If the value from F_{zk} is $(G,1)$, then simulate an interaction for V:
   (a) For all i, send $(\mathsf{sid}_i, \mathsf{receipt})$ to A.
   (b) Obtain the challenge c from A.
   (c) If $c=0$, then (by pretending to be F_{com}) send openings of a random permutation of G to A. If $c=1$, then (by pretending to be F_{com}) send an opening of a random Hamiltonian tour to A.

   In this case, the simulation is perfect.

2. If A controls the prover (weak extraction): S gets input z' from Z, and runs A on z'. Then:
   (a) Obtain from A all the commit messages to F_{com} and record the committed graph and permutation. Send $(\mathsf{sid}, P, V, G, h=\perp)$ to F_{wzk}.
   (b) Obtain the bit b from F_{wzk}. If $b=1$ (i.e., cheating is allowed), then send the challenge $c=0$ to A. If $b=0$ (no cheating allowed), then send $c=1$ to A.
   (c) Obtain A's openings of the commitments (either a permutation of the graph, or a Hamiltonian cycle). If $c=0$ (permutation) and all openings are consistent with G, then send $b'=1$ to F_{wzk}; if some openings are bad then send $b'=0$. If $c=1$ (cycle) and the openings are of a proper Hamiltonian cycle h' then send h' to F_{wzk}; otherwise send $h'=\perp$.

This simulation is also perfect: the challenge c is a uniform and independent bit, and V's output is 1 iff A opened all its commitments correctly (as determined by c).

3. The cases in which A controls both parties or neither party are trivial.

4. Handling adaptive corruptions is also trivial: neither party has any secret state, so the simulator need not provide anything special to A upon corruption.

□

2 From weak ZK to full ZK

The protocol for realizing F_{zk}^R in the F_{wzk}^R-hybrid model (for any relation R) is just what you'd expect: run k copies of F_{wzk} *in parallel*. The prover algorithm, on input (x, w), sends (x, w) to each of k copies of F_{wzk}. The verifier, on empty input, receives (x_i, b_i) from the ith copy. If all x_i are the same x and all b_i are the same b, output (x, b), else output $\perp$.

Claim 2 *Parallel composition of k copies of F_{wzk} realizes F_{zk}.*

Proof: Let A be an adversary in the F_{wzk}^R-hybrid model; we'll construct an adversary that interacts with F_{zk}^R and fools all environments. There are four cases:

1. If A controls the verifier: this case is simple. All A expects to see is k copies of (x, b) being delivered by the copies of F_{wzk}. S runs A, obtains (x, b) from F_{zk}^R, sends k copies to A, and outputs whatever A outputs.

2. If A controls the prover: here, A is providing k inputs $x_1, \ldots, x_k$ to the k copies of F_{wzk}^R, obtaining k bits $b_1, \ldots, b_k$ from these copies, and giving witnesses $w_1, \ldots, w_k$ and bits $b'_1, \ldots, b'_k$ in return. S runs A, obtains the x_is from A, chooses random bits $b_1, \ldots, b_k$ to give to A, and obtains the w_is and b'_is. Then:

   - If all the x_is are the same x and all copies of F_{wzk}^R would accept (either because $b_i = b'_i = 1$ or $R(x, w_i) = 1$ for all i), then pick a w_i such that $R(x, w_i) = 1$ and give (x, w_i) to F_{zk}^R. (If no such w_i exists, then fail; this can happen only when all b_i are 1, which occurs with probability 2^{-k}.)
   - Otherwise give $(x, \perp)$ to F_{zk}^R.

3. If A controls both parties or neither party, the simulation is trivial. Handling adaptive corruptions is trivial as well (no party has any secret state).

We analyze S: when the verifier is corrupted, the views of Z in the hybrid and ideal interactions are identically distributed. When the prover is corrupted, the only difference is that S may fail with probability 2^{-k}; conditioned on non-failure, the views are identical. Therefore the views are statistically indistinguishable. □

3 Realizing any two-party functionality

This section is based on [CLOS02], which is based on the paradigm of [GMW87]. The paradigm is to start with a protocol that is secure against *semi-honest* adversaries (i.e., those that always follow the protocol, even when corrupted). Then construct a general *compiler* that transforms protocols that are secure against semi-honest parties into protocols that are secure against malicious parties. We will first deal with two-party functionalities and then generalize to multi-party ones.

For the semi-honest case, we will proceed in three steps:

1. Present the ideal oblivious transfer (OT) functionality F_{ot}.

2. Show how to realize F_{ot} for semi-honest adversaries, in the plain model.

3. Show how to realize "any functionality" in the F_{ot}-hybrid model.

Before proceeding further, we note that there are two natural variants of semi-honest adversaries. In the first, the adversarial parties can change the inputs that we given by the environment, but are otherwise passive. In the second, the environment gives inputs directly to the parties, and the adversary merely listens (i.e., it cannot change the inputs). We will use the first variant to model semi-honest adversaries.

Here is the 1-out-of-m oblivious transfer functionality, F_{ot}^m (we name the sending party T to disambiguate with the simulator, and the receiving party R):

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

To realize F_{ot}^2, we can use the protocol from [EGL85]: let F be a family of trapdoor permutations and let B be a hard-core predicate for F. Then the protocol is the following:

1. $T(v_0, v_1)$ chooses f, f^{-1} from F and sends f to R.
2. $R(i)$, $i \in \{0, 1\}$, chooses x_0, x_1, and sets $y_i = f(x_i)$ and $y_{1-i} = x_{1-i}$, and sends (y_0, y_1) to T.
3. T computes $v_i \oplus B(f^{-1}(y_i))$ for $i = 0, 1$ and sends (t_0, t_1) to R.
4. R outputs $v_i = t_i \oplus B(x_i)$.

Claim 3 *The above protocol realizes F_{ot}^2 for* semi-honest *adversaries with* static *corruptions.*

Proof: For any A, we will construct an S that fools all Z. There are two interesting cases:

- A corrupts T: A expects to see T's input (v_0, v_1), plus the two values (y_0, y_1) received from R. S can easily simulate this view, where (v_0, v_1) are taken as T's inputs in the ideal process and (y_0, y_1) are random.
- A corrupts R: A expects to see R's input i, the function f, and the bits (t_0, t_1). S works as follows: obtain v_i from F_{ot}^2, give the ideal-process input i to A, select (f, f^{-1}) from F and give the pair to A, receive (y_0, y_1) from A where $y_i = f(x_i)$ and $y_{1-i} = x_{1-i}$ for random x_0, x_1 due to semi-honesty, and send (t_0, t_1) to A where $t_i = v_i \oplus B(x_i)$ and t_{1-i} is random.

We now analyze S. In the first case, the simulation is perfect. In the second case, the validity of the simulation reduces to the hardness of B: if we had an environment that could distinguish between real and ideal executions, we could distinguish between $(f(x_{1-i}), B(x_{1-i}))$ (the real-world case) and $(f(x_{1-i}), t_{1-i})$ (the ideal-world case), where x, t_{1-i} are random. □

Some remarks on the above protocol: it easy generalizes to n-out-of-m OT. To obliviously transfer k-bit strings, we can invoke the protocol k times (due to semi-honesty, the receiver will always ask for bits from the same string). For adaptive adversaries with erasures, the protocol can easily be made to work — R just erases x_0, x_1 before sending (y_0, y_1). Without erasures, we need to do something slightly different.

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

For preliminaries, we represent the functionality F as a Boolean circuit, with the following properties:

- F is a "standard functionality;" i.e. it has a "shell" and a "core," where the core does not know who is corrupted. Our protocol evaluates the core only.
- F is written as two-input $\oplus$ (addition mod 2) and $\wedge$ (multiplication mod 2) gates.
- The circuit has 5 types of input lines: inputs of P_0, inputs of P_1, inputs of S, random inputs, and local state inputs.

- The circuit has 4 types of output lines: outputs to P_0, ouputs to P_1, outputs to S, and local state for the next activation.

Now we describe the protocol in the F_{ot}-hybrid model:

1. **Share inputs.** When P_i is activated with a new input, it notifies P_{1-i} and:
   (a) Shares each input bit b with P_{1-i}, by sending $b_{1-i} \leftarrow \{0,1\}$ to P_{1-i} and keeps $b_i = b \oplus b_{1-i}$.
   (b) For each random input line r, chooses $r_0, r_1 \leftarrow \{0,1\}$ and sets r_{1-i} to P_{1-i}.
   (c) P_i and P_{1-i} keep the shares of the local state lines from the previous activation (initially, they are all set to 0).
   (d) P_i sets its shares of the adversary/simulator input lines to be 0.

   When P_i is instead notified by P_{1-i}, it proceeds as above except with input bits equal to 0.

   Now all inputs lines to the circuit have been shared between the two parties.

2. **Evaluate 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 be the inputs to the gate, and c denote the output.

   For an addition gate, each party just sums (mod 2) its shares of the two inputs.

   For a multiplication gate, P_0 and P_1 use F_{ot}^4 as follows: P_0 chooses c_0 at random, and acts as the sender with input

$$\begin{aligned} v_{00} &= a_0 b_0 + c_0 \\ v_{01} &= a_0(1-b_0) + c_0 \\ v_{10} &= (1-a_0) b_0 + c_0 \\ v_{11} &= (1-a_0)(1-b_0) + c_0 \end{aligned}$$

   while P_1 plays the receiver with input (a_1, b_1) and sets the output to be c_1. It is easy to verify that $c_0 \oplus c_1 = (a_0 \oplus a_1)(b_0 \oplus b_1)$.

3. **Generate outputs.** Once all the gates have been evaluated, each output has been 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 the values on its output lines and outputs them; P_i keeps its share of each local-state line to be used in the next activation; outputs to the adversary *are ignored*.

Claim 4 *Let F be any standard ideal functionality. Then the above protocol realizes F in the F_{ot}-hybrid model for semi-honest, adaptive adversaries.*

Proof Sketch: For any A, we construct an S that fools all Z. The simulation will be *unconditional* and *perfect*. Here is why and how:

- 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 trivial for S to simulate.
- P_1 receives the corresponding information, plus random shares of all intermediate values from F_{ot}. This is also easy to simulate.
- Upon corruption, it's easy to generate local state that is consistent with the protocol.

□

Some remarks: there is a protocol by Yao that works in a constant number of rounds, which can be proven secure for static adversaries, and also can be made to work, with erasures, against adaptive adversaries.

Research Question 1 *Without erasures, is there a general, constant-round construction?*

3.2 Protocol compilation

In the spirit of [GMW87], our aim is to force malicious parties to follow the semi-honest protocol specification. Here's how:

- Parties should *commit* to their inputs.
- Parties should *commit* to *uniform* random tapes (using secure coin-tossing to ensure uniformity).
- Run the semi-honest protocol Q, and in addition, parties use *zero-knowledge protocols* to prove that they have been following Q. That is, each message m of Q is followed by a proof of the NP statement: "there exist input x and random input r that are the legitimate openings of the commitments above, and such that the message m is the result of running the protocol on x, r, and the messages I received so far."

Consider the construction of a UC "GMW compiler." The naive approach is to construct such a compiler, given access to the ideal commitment and ZK functionalities, compose with protocols that realize those functionalities, and use the composition theorem to prove security. However, if ideal commitment is used, there is no commitment string to prove statements about! This calls for a new "commit and prove" primitive that combines the two functionalities: parties should be able to commit to values, and prove, in zero-knowledge, statements about those values. Here is the formal definition of the F_{cp}^R functionality for a given poly-time relation R:

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

Some remarks about this functionality: V is assured that the value x it receives is in the relation R. P is assured that V learns nothing other than x and $R(x, W)$. In the F_{cp}-hybrid model, we can do the GMW compiler *without computational assumptions!* (See below.)

Here is a protocol to realize F_{cp}^R in the F_{zk}-hybrid model. The protocol uses Com, which is any perfectly binding, non-interactive commitment scheme.

1. On input $(\mathsf{sid}, C, V, \mathsf{commit}, w)$ [i.e., to commit to value w], C computes $a = \text{Com}(w, r)$ for random r, adds w to the list W, adds a to the list A, adds r to the list R, and sends $(\mathsf{sid}, C, V, \mathsf{prove}, a, (w, r))$ to $F_{zk}^{R_c}$, where $R_c = \{(a, (w, r)) \ : \ a = \text{Com}(w, r)\}$.
2. Upon receiving $(\mathsf{sid}, C, V, a, 1)$ from $F_{zk}^{R_c}$, V adds a to its list A, and outputs $(\mathsf{sid}, C, V, \mathsf{receipt})$.
3. On input $(\mathsf{sid}, C, V, \mathsf{prove}, x)$ [i.e., to prove statement x], C sends $(\mathsf{sid}, C, V, \mathsf{prove}, (x, A), (W, R))$ to $F_{zk}^{R_p}$, where

$$R_p = \{((x, A), (W, R)) \ : \\ W = w_1 \cdots w_n, A = a_1 \cdots a_n, R = r_1 \ \cdots r_n, R(x, W) = 1 \text{ and } a_i = \text{Com}(w_i, r_i) \text{ for all } i.\}$$

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

Theorem 1 *The above protocol realizes F_{cp}^R in the F_{zk}-hybrid model for* non-adaptive adversaries*, and assuming the security of* Com.

Proof: For any A, we construct an S that fools all Z. First, S runs A. Then we consider two cases:

- The committer is corrupted: in the commit phase, S obtains from A the message $(\mathsf{sid}, C, V, \mathsf{prove}, a, (w, r))$ to $F_{zk}^{R_c}$. If R_c holds on a and (w, r), then S sends $(\mathsf{sid}, C, V, \mathsf{commit}, w)$ to F_{cp}. In the proof phase, S obtains from A the message $(\mathsf{sid}, C, V, \mathsf{prove}, (x, A), (W, R))$ to $F_{zk}^{R_p}$. If R_p holds on (x, A) and (W, R), then S sends $(\mathsf{sid}, C, V, \mathsf{prove}, x)$ to F_{cp}.

- The verifier is corrupted: in the commit phase, S obtains from F_{cp} a $(\mathsf{sid}, C, V, \mathsf{receipt})$ message, and sends to A the message (sid, C, V, a), where $a = \text{COM}(0, r)$ for random r. In the proof phase, S obtains from F_{cp} a (sid, C, V, x) message, and simulates for A the message $(\mathsf{sid}, C, V, (x, A))$ from $F_{zk}^{R_p}$, where A is the list of commitments that S has generated so far.

Let's analyze S: for a corrupted committer, the simulation is perfect. For a corrupted verifier, the only difference between simulated and real executions is that the commitments are all to 0 instead of the witnesses. If Z distinguishes between these two cases, it's straightforward to use Z to break the hiding property of the commitment scheme. □

Some remarks: the simulation fails in case of adaptive adversaries, even with erasures, because the commitments are perfectly binding (and hence cannot correspond to the good witnesses). We can fix this problem by using "equivocable commitments."

Research Question 2 *Can F_{cp} be realized unconditionally (i.e., in some hybrid model without computational assumptions)?*

Now that we have F_{cp}, we can construct the full protocol (in the F_{cp}-hybrid model) for malicious adversaries. Let $P = (P_0, P_1)$ be the protocol that assumes semi-honest adversaries. The protocol $Q = C(P)$ uses two copies of F_{cp}, where in the ith copy, Q_i is the prover and Q_{1-i} is the verifier. The code for Q_0 is as follows (and the code for Q_1 is similar):

1. **Commit to Q_0's randomness.** Q_0 chooses random r_0 and sends $(\mathsf{sid}.0, Q_0, Q_1, \mathsf{commit}, r)$ to F_{cp}. Q_0 receives r_1 from Q_1, and sets $r = r_0 \oplus r_1$.

2. **Commit to Q_1's randomness.** Q_0 receives $(\mathsf{sid}.1, Q_1, Q_0, \mathsf{receipt})$ from F_{cp} and sends a random value s_0 to Q_1.

3. **Receive the input x in the ith invocation.** Q_0 sends $(\mathsf{sid}.0, Q_0, Q_1, \mathsf{commit}, x)$ to F_{cp}. Let M be the list of messages seen so far. Q_0 runs the protocol P on input x, random input r, and messages M, and obtains either:

   - A local value, in which case Q_0 outputs this value.
   - An outgoing message m. In this case, send $(\mathsf{sid}.0, Q_0, Q_1, \mathsf{prove}, m)$ to F_{cp}, where the relation used by F_{cp} is:
   $$R_p = \{((m, M, r_2), (x, r_1)) \ : \ m = P_0(x, r_1 \oplus r_2, M)\}$$

4. **Receive the ith message m.** Q_0 receives $(\mathsf{sid}.1, Q_1, Q_0, \mathsf{prove}, (m, M, s_1))$ from F_{cp}. Q_0 verifies that s_1 is the value it sent in Step 2, and that M is the list of messages it has sent to Q_1. If so, then Q_0 runs P_0 on incoming message m and continues as in Step 3.

Theorem 2 *Let P be any two-party protocol. Then the protocol $Q = C(P)$, run with malicious adversaries in the F_{cp}-hybrid model, emulates protocol P, when run with semi-honest adversaries.*

Formally: for any malicious adversary A there exists a semi-honest adversary S such that for any environment Z we have:

$$\mathsf{EXEC}_{P,S,Z} \approx \mathsf{EXEC}^{F_{cp}}_{Q,A,Z}.$$

Corollary 1 *If protocol P securely realizes F for semi-honest adversaries then $Q = C(P)$ securely realizes F in the F_{cp}-hybrid model for malicious adversaries.*

Proof Sketch: We will skip the details of the proof, since they are pretty straightforward. However, we highlight a few of the properties of the proof: it is unconditional, and offers perfect simulation. It works even for adaptive adversaries. However, it requires S to be able to change the inputs of the semi-honest parties (hence our choice of the specific semi-honest model above). □

4 Extending to the multi-party case

There are a number of challenges we must conquer in order to to multi-party computation:

- Construct a basic, semi-honest protocol.
- Deal with asynchronous channels with no guaranteed message delivery.
- Deal with issues of broadcast/Byzantine agreement.
- Figure out how and where to use existing primitives (OT, commitment, ZK, commit-and-prove).
- Deal with a variable number of parties.

References

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). *STOC 1988*, pages 1–10, 1988.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. *STOC 2002*, pages 494–503, 2002.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC 1987*, pages 218–229. ACM, 1987.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multi-party protocols with honest majority (extended abstract). *STOC 1989*, pages 73–85, 1989.