CS 65500
Advanced Cryptography

Lecture 18: GMW Compiler

Instructor: Aarushi Goel

Spring 2025

Agenda

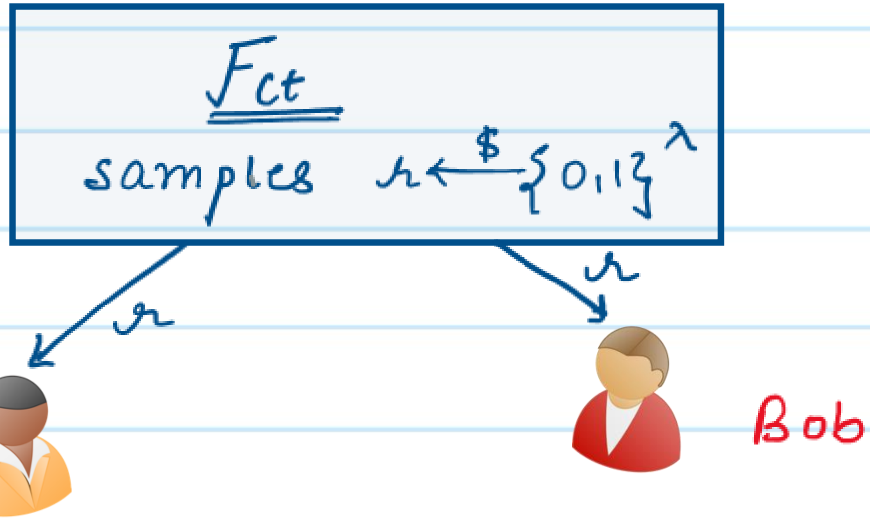
- → Coin Toss
- → GMW Paradigm: Malicious Security with abort

Reminder: HW 5 will be released tonight!

Two-Party Coin Toss

→ A secure two-party coin tossing protocol enables two-mutually distrusting parties to obtain unbiased random strings.

→ In other words, it is a two-party protocol that securely realizes the following functionality in the presence of a malicious adversary:



Observe that this is an input-less functionality!

Candidate Construction for Two-Party Coin Toss

Alice 

 Bob

Alice:

$r_1 \xleftarrow{\$} \{0,1\}^\lambda$

$s \xleftarrow{\$} \{0,1\}^\lambda$

Alice $\xrightarrow{c = Com(r_1; s)}$ Bob

Bob:

$r_2 \xleftarrow{\$} \{0,1\}^\lambda$

Alice $\xleftarrow{r_2}$ Bob

Alice $\xrightarrow{r_1,\, s}$ Bob

Alice: Output $r = r_1 \oplus r_2$

Bob: If $c = Com(r_1; s)$, output $r = r_1 \oplus r_2$

This protocol is not secure!

→ The simulator given a random r from fct is now unable to fix r_1 such that $r_1 \oplus r_2 = r$, since r_2 depends on r_1

A Secure Coin-Tossing Protocol

Alice 

 Bob

$r_1 \xleftarrow{\$} \{0,1\}^\lambda$

$s \xleftarrow{\$} \{0,1\}^\lambda$

$\xrightarrow{c = Com(r_1; s)}$

a ZKPoK that Alice knows r_1, s; such that $c = com(r; s)$

$r_2 \xleftarrow{\$} \{0,1\}^\lambda$

$\xleftarrow{r_2}$

$\xrightarrow{r_1}$

a ZKP that c is a commitment to r_1

Output $r = r_1 \oplus r_2$

output $r = r_1 \oplus r_2$

Security Against Malicious Bob.

A simulator S^{B^*} for Bob will proceed as follows:

1 Query F_{ct} to get r
2. Compute $c = Com(0;s)$ & send it to B^*
3. Simulate ZKPoK about validity of c.
4. Receive r_2 from B^*
5. Send $r_1 = r \oplus r_2$ to B^*
6. Simulate the ZKP that initial commitment was to r_1.

Security Against Malicious Bob.

We can use the following sequence of hybrids to show indistinguishability between the simulated transcript & Bob's view in the real protocol:

H_0 Bob's view in the Real protocol

H_1 Simulate ZKPoK about validity of c

H_2 Simulate the ZKP that initial commitment was to r_1

H_3 Compute $c = Com(0;s)$ & send it to B^*.

H_4 Simulated transcript

Security Against Malicious Alice.

A simulator S^{A^*} for Alice will proceed as follows:

1 Query F_{ct} to get r
2. Receive a commitment c & ZKPoK from A^*.
3. Verify ZKPoK and extract r_1.
4. Send $r_2 = r \oplus r_1$ to A^*
5. Receive r_1' & ZKP from A^*
6. Check if $r_1 = r_1'$ & verify ZKP w.r.t. r_1

Security Against Malicious Alice.

We can use the following sequence of hybrids to establish indistinguishability between the simulated transcript and Alice's view in the real protocol:

H_0 Alice's view in the real protocol

H_1 Verify ZKPoK and extract r_1

H_2 Check if $r_1 = r_1'$, if not output $\perp$ and terminate

H_3 Verify ZKP w.r.t. r_1

H_4 Simulated transcript.

Malicious Security with Abort

→ This coin tossing protocol only achieves *security with abort* against a malicious adversary

→ In other words, the adversary can cause the protocol to abort, preventing the honest party from learning the output.

→ However, in case the adversary does not abort & both parties learn the output, then the output is guaranteed to be an unbiased random value.

Maliciously Secure MultiParty Computation for General Functions

→ This approach of commiting to messages and then attaching zero-knowledge proofs can be generalized to transform any semi-honest secure multiparty computation protocol into one that achieves security with abort against malicious adversaries.

→ This approach was first introduced by Oded Goldreich, Silvio Micali and Avi Wigderson.

GMW Compiler

MPC protocol secure against semi-honest adversaries

↓ coin tossing

MPC protocol secure against semi-malicious adversaries

↓ commitments + ZKPOK

MPC protocol secure (with abort) against malicious adversaries

Maliciously Secure Two-Party Computation Protocol

Let Π_{sh} be a semi-honest secure protocol for computing a function f between Alice and Bob.

* <u>Input Commitment Phase</u> :

Alice  (x)

 Bob (y)

Alice → Bob: com(x)

Alice ↔ Bob: A ZKPoK proving that Alice knows x that is consistent with this commitment

Bob → Alice: com(y)

Alice ↔ Bob: A ZKPoK proving that Bob knows y that is consistent with this commitment

* Coin Tossing Phase : In this phase, the two parties engage in secure coin tossing protocols, where one party receives a commitment to a random string and the other party receives the string itself plus the decommitment of the string.

Alice   Bob

A modified coin tossing protocol
where Alice obtains a random string r_A,
$\longleftrightarrow$
$com(r_A; s_A)$ and s_A, while Bob gets $com(r_A; s_A)$

A modified coin tossing protocol
where Bob obtains a random string r_B,
$\longleftrightarrow$
$com(r_B; s_B)$ and s_B, while Alice gets $com(r_B; s_B)$

* Protocol Emulation: Alice and Bob run the semi-honest protocol Π_{sh} with inputs x, y resp. and random tapes r_A, r_B resp. Additionally, along with every message they prove using zero-knowledge proofs that these messages where consistent with input (committed to during the input commitment phase) and the random tape (obtained during the coin tossing phase).