CS 65500
Advanced Cryptography

Lecture 17: Coin Toss

Instructor: Aarushi Goel

Spring 2025

Agenda

- → Commitments
- → Coin Toss

Defining Interactive Proofs (without Zero-Knowledge)

Definition: A protocol π between a prover P and a verifier V is an interactive proof system for a language L if V is a PPT machine and the following properties hold:

- Completeness: $\forall x \in L$

$$\Pr\left[\mathrm{Out}_V\left[P(x) \longleftrightarrow V(x)\right] = 1\right] = 1$$

- Soundness: There exists a negligible function $\nu(.)$, s.t., $\forall x \notin L$, $\forall \lambda \in \mathbb{N}$ and all adversarial provers P^*,

$$\Pr\left[\mathrm{Out}_V\left[P^*(x) \longleftrightarrow V(x)\right] = 1\right] \leq \nu(\lambda)$$

We can also modify the above definition to consider PPT provers

Proofs that are only sound against PPT provers are called arguments.

Defining Zero-Knowledge

Definition: An interactive proof Π between P & V for a language L with witness relation R is said to be zero-knowledge if for every n.u. PPT adversary V^*, there exists a (expected) PPT simulator S, such that $\forall x \in L, \forall w \in R(x), \forall z \in \{0,1\}^*$ and $\forall \lambda \in \mathbb{N}$, the following two distributions are computationally indistinguishable:

1. $\{\text{View}_{V^*}[P(x,w) \longleftrightarrow V^*(x,z)]\}$
2. $\{S^{V^*}(1^\lambda, x, z, L)\}$

We can also consider the notions of statistical / perfect zero-knowledge against unbounded adversaries, if the above distributions are statistically close (or identical respectively)

Defining Zero-Proofs of Knowledge

Definition: A zero-knowledge proof π between P & V for a language L, with witness relation R_L is said to be a proof of knowledge with knowledge error ϵ, if $\exists$ an algorithm E^{P^*}, called an extractor, that runs in expected polynomial time, such that the following holds for every x and every P^*

$$\Pr[\mathrm{Out}_V[P^*(x) \longleftrightarrow V(x)] = 1] - \Pr[R_L(x, w) = 1 \quad w \leftarrow E^{P^*}(x)] \leq \epsilon$$

ZKPs that only satisfy knowledge soundness against PPT provers are called <u>arguments of knowledge</u>

Defining Maliciously Secure MPC

Definition: A protocol π securely realizes $\mathcal{F}$ in the presence of malicious adversaries, if $\exists$ a PPT simulator algorithm Sim, such that $\forall$ PPT malicious adversaries A corrupting any t-sized subset $C \subset [n]$ of the parties, $\forall \lambda \in \mathbb{N}$ and $\forall \{x_i\}_{i \notin C}$, the following two distributions are computationally / statistically / perfectly indistinguishable:

$$\text{Real}_{\pi, A}(\lambda, \{x_i\}_{i \notin C})$$

$$\text{Ideal}_{\mathcal{F}, \text{Sim}^A}(\lambda, \{x_i\}_{i \notin C})$$

Formalizing the Requirements of a Maliciously Secure MPC in the Real-Ideal World Paradigm

Let $C \subset [n]$ be t-sized subset of corrupt parties. We define two distributions:

1 $\text{Real}_{\pi,A}(\lambda, \{x_i\}_{i \notin C})$: Run the protocol using λ as the security parameter & $\{x_i\}_{i \notin C}$ as the inputs of the honest parties. The messages of corrupt parties are chosen based on A. Let y_i denote the output of each honest party P_i & View_i denote the view of each P_i in this protocol.

Output: $\{\{\text{view}_i\}_{i \in C}, \{y_i\}_{i \notin C}\}$

Let Sim^A be a PPT algorithm given oracle access to A.

2. $\text{Ideal}_{F, Sim^A}(\lambda, \{x_i\}_{i \notin C})$: Run Sim^A until it outputs $\{x_i\}_{i \in C}$, compute $(y_1, \ldots, y_n) \leftarrow F(x_1, \ldots x_n)$. Then, give $\{y_i\}_{i \in C}$ to Sim^A. Let $\{view^*_i\}_{i \in C}$ denote the final output of Sim^A.

Output: $\{ \{view^*_i\}_{i \in C}, \{y_i\}_{i \notin C} \}$

Commitments

→ Commitments are a digital analogue of locked boxes

→ Comprise of two phases:

* Commit phase: Sender locks a value v inside the box

* Open phase: Sender unlocks the box to reveal v.

→ Properties that we need from a commitment scheme:

* Hiding: Contents of the box remain hidden from the receiver until it is unlocked by the sender

* Binding: Once the sender locks the box and sends it to the receiver the sender can no longer change its contents

Defining Commitments

Definition: A randomized polynomial time algorithm Com is called a commitment scheme for n-bit strings if it satisfies the following properties:

* Binding: $\forall\, v_0, v_1 \in \{0,1\}^n$ and $r_0, r_1 \in \{0,1\}^n$, it holds that $Com(v_0; r_0) \neq Com(v_1; r_1)$
* Hiding: $\forall\, v_0, v_1 \in \{0,1\}^n$, the following distributions are computationally indistinguishable:
    - $\{ r \xleftarrow{\$} \{0,1\}^n ;\ Com(v_0; r) \}$
    - $\{ r \xleftarrow{\$} \{0,1\}^n ;\ Com(v_1; r) \}$

Construction of Bit commitments

The following scheme can be used for commiting to bits.

Let f be a one-way permutation, h be the hard core predicate for f.

* Commit Phase: Sender computes $Com(b; r) = f(r), b \oplus h(r)$. Let C denote this commitment.

* Open Phase: Sender reveals (b, r). Receiver accepts if $C = (f(r), b \oplus h(r))$, and rejects otherwise.

Security:

- Binding holds because f is a permutation
- Hiding follows from the property of hard-core predicates.

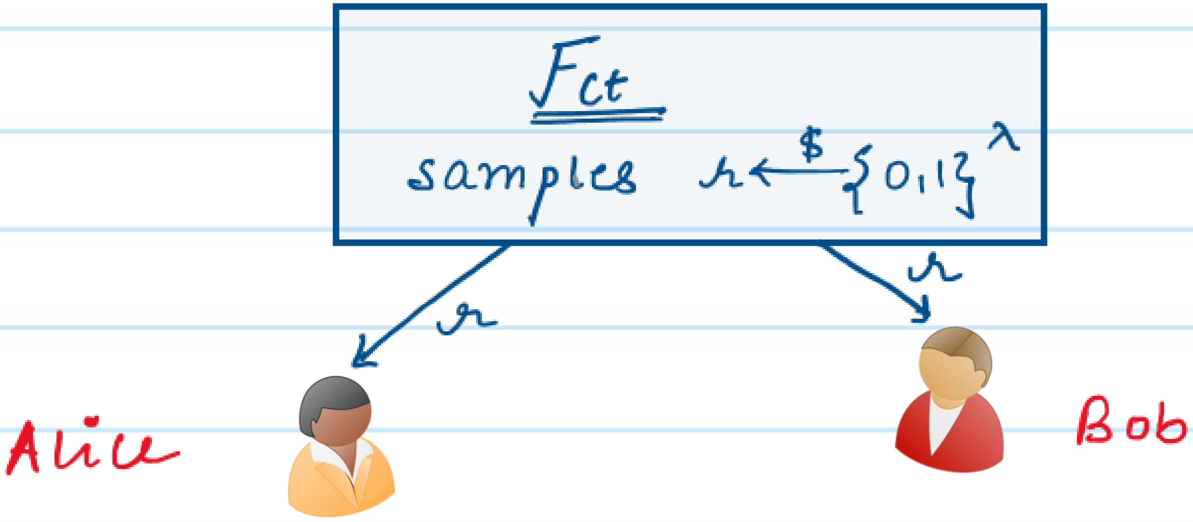
(Think of a formal proof)

Multibit commitment: How can one go from single bit commitment to a multi-bit commitment?

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

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

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

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

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

$\xleftarrow{r_2}$

$\xrightarrow{r_1, s}$

Output $r = r_1 \oplus r_2$

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 f_{ct} 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 $c = \mathrm{Com}(0; s)$ & 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