ECE598 Ideal Functionalities, Spring 2022
Lec 1: Introduction to UC

Lecturer: Andrew Miller
Scriber: Mingjia Huo (mhuo4)

Date: Jan18, 2022

As the first lecture in this semester, we introduce the model of Ideal and Real Paradigm with an example of Secure 2-party Auction. To define security in UC, we define four components: environment, protocol, adversary, and functionality. The main security paradigm is to show any attack in the real world was already possible in the ideal world. We give the model of Commitment and we will show its security in future lectures. We also give a brief introduction to the Composition Theorem.

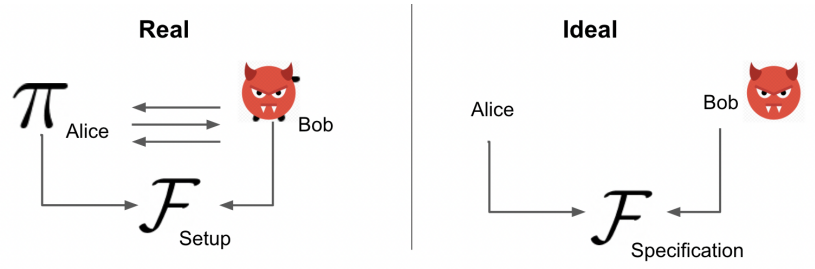
1 Course Overview

- Lectures:
  - Introduction to UC lectures;
  - Practice writing UC proofs;
  - Getting started using Python-SaUCy framework tools.
- Mix of lectures/discussion, student presentations:
  - In depth analysis of the UC framework
  - Tour of Cryptography constructions in UC, including ZK, MPC, Blockchains
- Discussions and student presentations
  - Extensions to the framework
  - Interesting new proof techniques (Algebraic Group Model + UC)
  - Formal verification tools and UC (EasyUC, IPDL)
  - Applications of your interest

- Refer to course slices for course logistics, including scribing notes, presentations, course projects and references. Also set up SaUCy, a programming framework for UC.

2 Introduction to Ideal/Real Security

Here is an illustration for Ideal/Real Paradigm.



Real World

- Protocols: Codes for multiple parties.
- Adversary: Some parties may be corrupted, in which case the Adversary controls them (static byzantine model).
  Static byzantine: the corrupted parties have to be decided at the beginning of the protocol and will be controlled by the adversary.
- $\mathcal{F}$: The parties will make use of an ideal functionality $\mathcal{F}$, to model network assumptions, existing primitives, trusted setups, etc.

Ideal World

- Parities: There's no protocol. Parties will delegate the work directly to the Idealized service.
- Adversary: An attacker in the ideal world can only do what the Ideal Functionality indicates.
- $\mathcal{F}_{\text{Specification}}$: We use an Ideal Functionality $\mathcal{F}_{\text{Specification}}$ to give the specification for our desired protocol. It describes:

A program for a trusted third party that behaves just like our protocol.

$\mathcal{F}_{\text{Specification}}$ may or may not know the corrupted parties and treat them differently from honest parties. It depends on model settings.

Ideal/Real Paradigm Protocols in the real world meet the specification if it's "just as good as the ideal". That is,

Any attack on the Real world was already possible in the Ideal world too.

2.1 Example: Secure 2-Party Auction

Specfication (Ideal World)

- Alice and Bob submit X and Y secretly to $\mathcal{F}_{\text{SecureAuction}}$ respectively.
- $\mathcal{F}_{\text{SecureAuction}}$: (1). Check X,Y balances. (2). Transfer to max(X,Y). (3). Refund to min(X,Y).

Implementation using Yao's Garbled Circuits protocol (Real World)

- Protocols π: Two parties use Yao-Generator and Yao-Evaluator and compute cryptography locally. They will communicate with $\mathcal{F}_{\text{OT,BB}}$ (OT=Oblivious Transfer, BB=Bulletin Board)
- $\mathcal{F}_{\text{OT,BB}}$: Parties exchange messages by posting on public Bulletin Board, and make use of Oblivious Transfer primitive.

In both worlds, we will have an environment $\mathcal{Z}$, which is an honest party to provide protocol inputs and receive outputs.

3 Defining Security in UC

Any attack on the Real world was already possible in the Ideal world too.

To defining security in this model, we can follow the steps below:

1. Write down the application specification and the starting assumptions as Ideal Functionalities.

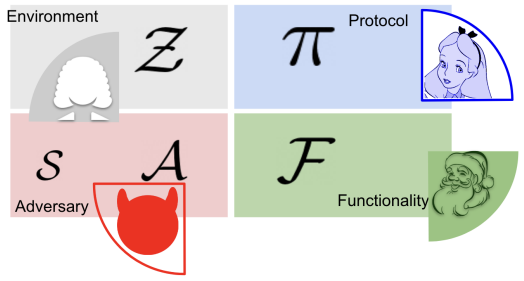
2. Define the protocol P.

3. Construct a simulator S for the dummy adversary.

4. Show the simulation holds for every environment. That is

$$\forall \mathcal{A}, \exists S, s.t. \forall \mathcal{Z}, Real_{\mathcal{Z},P,\mathcal{A},F_{\text{Start}}} \sim Ideal_{\mathcal{Z},id,S,F_{Goal}},$$

The following figure shows the basic components in the UC framework.



3.1 Example: Commitment

Ideal World Note that there are no protocols in the ideal world. The environment will directly communicate with the ideal functionality $\mathcal{F}_{\text{Com}}$.

$\mathcal{F}_{\text{Com}}$(**Sender**, **Receiver**)

On input (Commit, b) from Sender:
  Record b
  Send (Commited) to Receiver
On input (Open) from Sender:
  Send (Opened, b) to Receiver

Real World The protocols are as follows. Security will be discussed in future lectures.

$\mathsf{Prot}_{\mathsf{Com}}(\textbf{Sender}, \textbf{Receiver})$

As Sender:
On input (Commit, b) from $\mathcal{Z}$:
  Sample $r \leftarrow 2^k$
  $h \leftarrow$ query $\mathcal{F}_{\text{RandomOracle}}(r \parallel b)$
  Send (Commit, h) to Receiver
On input (Open) from $\mathcal{Z}$:
  Send (Opening, b, r) to Sender

As Receiver:
On input (Commit, h) from Sender:
  Record h
  Output (Committed) to $\mathcal{Z}$
On input (Opening, b, r) from Sender:
  $h' \leftarrow$ query $\mathcal{F}_{\text{RandomOracle}}(r \parallel b)$
  Verify $h' = h$
  Output (Opened, b) to $\mathcal{Z}$

4 Modular Protocol Composition

- Start with an application, modeled as an ideal functionality.
- Application makes use of some generic underlying cryptographic primitive, an abstraction layer.
- Finally, the generic cryptographic primitive is instantiated based on some network assumptions, setup, or other primitives.

Composition Theorem ensures that the composed protocol is secure as long as the individual layers are proven secure.