6.897: Selected Topics in Cryptography 27 February 2004

Lecture 8: The Dummy Adversary

Instructor: Ran Canetti *Scribed by: Jonathan Herzog*

1 Previous Lecture

- Motivation for the Universally Composable (UC) framework
- Definition of an interactive Turing Machine (ITM) system with a variable number of machines
- Definition of the UC framework

2 The Dummy Adversary

In this lecture, we consider a variant definition of security that uses a fixed adversary A_D called the "dummy" adversary. This adversary is simply a proxy for the environment:

- All communication received from the protocol participants are fowarded directly to the environment Z. That is, when A_D receives incoming message m from party p, it simply writes "received m from p" on the subroutine tape of Z.
- It sends messages from the adversary to the parties, as directed. That is, when A_D receives "send m to p" from Z, it writes m on the incoming message tape of party p.

Why are we interested in this dummy adversary? Because security against this adversary is sufficient to ensure security in the general UC framework. That is:

Definition 1 *We say that a protocol* Π *realizes functionality* $\mathbf{F}$ *w.r.t. the dummy adversary* A_D *if there exists an ideal-process simulator* S *such that for all environments* Z*:*

$$\mathbf{Ideal}^{\mathbf{F}}_{\mathsf{S},\mathsf{Z}} \approx \mathbf{Exec}_{\Pi,\mathsf{A}_D,\mathsf{Z}}$$

Theorem 1 *Protocol* Π *realizes* $\mathbf{F}$ *with respect to the dummy adversary* A_D *iff it realizes* $\mathbf{F}$ *(according to the standard UC definition).*

Proof. One direction is simple: if Π realizes $\mathbf{F}$ in the standard UC framework, then for *every* adversary there exists a simulator S which is indistinguishable to every environment Z. Hence, there must be such a simulator for the dummy adversary A_D.

The other direction is more complex, and will not be proven via the contrapositive! We know that Π achieves functionality $\mathbf{F}$ with respect to the dummy adversary A_D. Hence, there exists a simulator S_D which is indistinguishable from the dummy adversary to every environment Z.

Now, suppose that the environment is communicating with any other adversary A. Then we construct a simulator for A as in Figure 1. That is, the simulator S operates as follows:

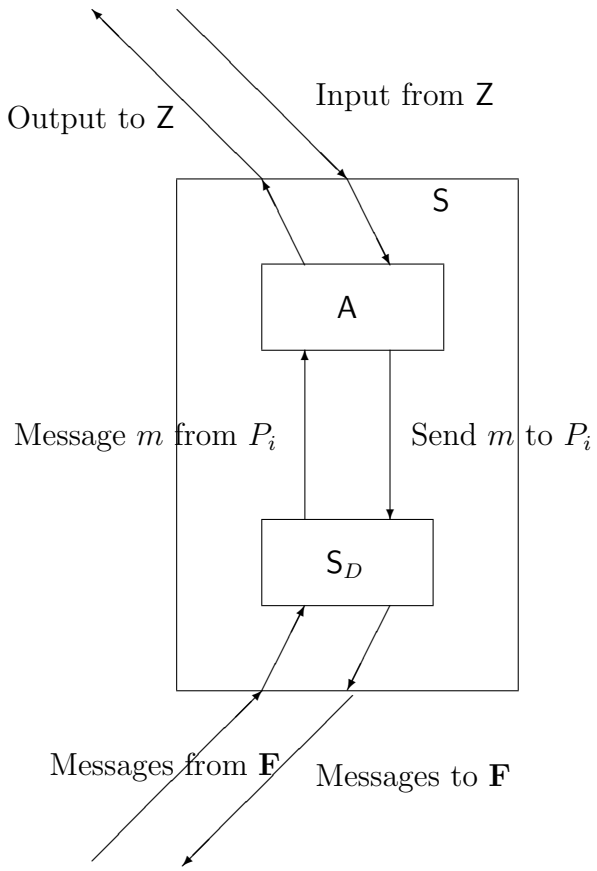


Figure 1: The Simulator S

- It simulates both the adversary A and the dummy simulator S_D.
- Messages from the environment go right to A.
- Messages from A go to the environment.
- Messages from S_D to the functionality are actually forwarded to the functionality, and vice-versa.
- When A sends a message m to party P_i, the message is sent to S_D, and messages from S_D ostensibly from party P_i are sent to the adversary A.

Now, will this simulator be indistinguishable from A to all environments in the real system? We show the answer to be "yes." For a given environment Z and adversary A, we construct an environment Z_D as in Figure 2 (ignoring the dashed box for the moment). That is, Z_D operates as follows:

- Z_D simulates both the environment Z and the adversary A.
- The input to Z_D is given to the simulated Z, and the output from Z is given as the output of Z_D.

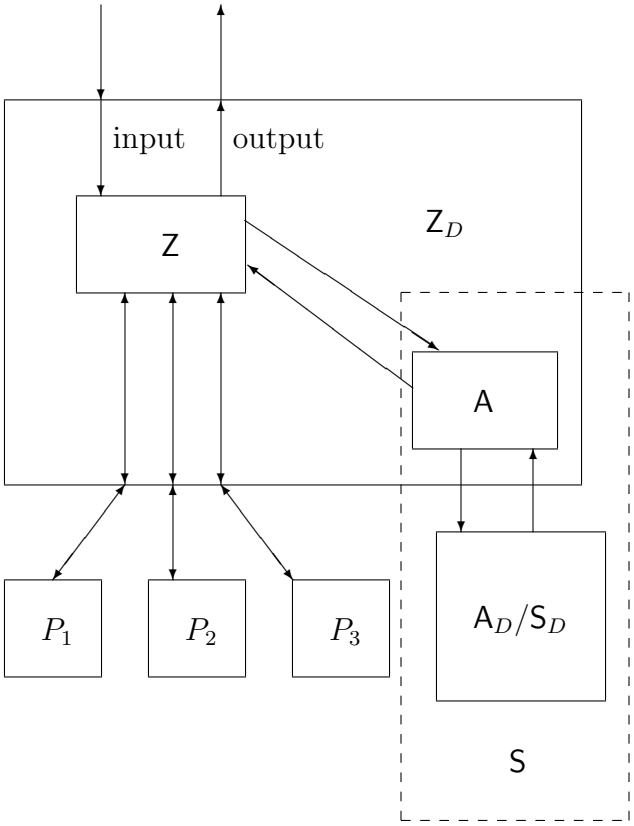


Figure 2: The environment Z_D

- Messages from the simulated environment Z to parties are forwarded to the parties, and messages from the parties to Z_D are forwarded to the simulation of Z.
- Messages from the simulated Z to the adversary are given to the simulated A, and messages from the simulated A to the environment are given to the simulated Z.
- Messages from the simulated adversary to parties are *not* given to the real parties, but sent to the real adversary. (In the diagram, we've already replaced this "real" adversary with either A_D or S_D.)
- Likewise, messages from the real adversary are given to the simulated A.
- Messages between the parties and real adversary are forwarded and returned as usual. Also, messages between the real simulator and functionality are passed and returned as usual.

By definition, S_D is indistinguishable from A_D to all environments, so it is indistinguishable to this one in particular. Now, look what happens when we give either A_D or S_D to the dummy environment Z_D as the real adversary (as indicated in the diagram above).

First, notice that the output of Z_D with A_D running protocol Π will be indistinguishable from the output of Z_D with S_D and functionality $\mathbf{F}$:

$$\mathbf{Exec}_{\Pi,\mathsf{A}_D,\mathsf{Z}_D} \approx \mathbf{Ideal}^{\mathbf{F}}_{\mathsf{S}_D,\mathsf{Z}_D}$$

However, ignore the solid box for Z_D and notice that the dashed box is S. Then we see that the combination of Z_D, S_D and $\mathbf{F}$ is exactly the same as the combination of the real environment Z, functionality $\mathbf{F}$, and the simulator constructed above for the arbitrary adversary A:

$$\mathbf{Ideal}^{\mathbf{F}}_{\mathsf{S}_D,\mathsf{Z}_D} = \mathbf{Exec}_{\mathbf{F},\mathsf{S},\mathsf{Z}}$$

(Note the strict equality.) Likewise, the combination of Z_D with dummy adversary A_D in an execution of Π is exactly the same as the combination of Z, protocol Π, and a composite adversary composed of A and A_D. However, the dummy adversary A_D simply forwards messages, so this composite adversary is equivalent to A. Hence:

$$\mathbf{Exec}_{\Pi,\mathsf{A}_D,\mathsf{Z}_D} = \mathbf{Exec}_{\Pi,\mathsf{A},\mathsf{Z}}$$

Hence,

$$\mathbf{Ideal}^{\mathbf{F}}_{\mathsf{S},\mathsf{Z}} \approx \mathbf{Exec}_{\Pi,\mathsf{A},\mathsf{Z}}$$

Note that it is essential for this proof that the dummy adversary only forward messages, and that we assume S_D accurately simulates A_D for protocol Π/functionality $\mathbf{F}$. Lastly, this theorem crucially depends on the possibility of arbitrary communication between environment and adversary, so that we can think of A as being both part of environment Z_D and outside of Z. □

3 Hybrid execution

In this section, we define the hybrid execution model which will be used in the main UC theorem. This execution model is much like previous ones, except that now parties can access multiple copies of the functionality $\mathbf{F}$. In particular, the hybrid model of protocol Π with ideal functionality $\mathbf{F}$ and environment Z is system of ITMs as follows:

- The initial TM is the distinguished machine Z, and it has a fixed ID.
- The initial TM Z can activate a single machine to act as the adversary A.
- It can also activate many copies of other ITMs to act as the honest parties. These parties run protocol Π and must all have the same session ID *sid*.
- The adversary A can send messages to the incoming message tapes of all parties and to subroutine tape of the environment.
- All honest parties can write messages on the incoming message tape of A. They can invoke new parties to run Π. They can write to the subroutine tape of their invoker, and can write to the input tapes of those machines they invoke.
- All honest parties (and the adversary) can send messages to and receive messages from *multiple* copies of functionality $\mathbf{F}$.

  - Each copy of $\mathbf{F}$ has a unique identifier *sid* and a number of dummy parties associated with it.
  - To interact with a copy of $\mathbf{F}$ with identifier *sid*, a party playing the role *pid* sends a message to the dummy party with $ID = (sid, pid)$. Likewise, a message from the functionality is sent to the dummy party with $ID = (sid, pid)$ (for some *pid*).

- The adversary A can write a special "corrupt" message on the incoming message tape of any party P. When this happens:
  - The party P writes "corrupted" on the subroutine tape of Z.
  - At each subsequent activation, P sends its entire state to Z.
  - A assumes all *write* privileges of P.
  - The reaction of $\mathbf{F}$ to corruption messages is left up to the discretion of the definer of $\mathbf{F}$. Such messages are also written on the subroutine tape of Z, however.

4 The Composition Operation

Suppose that Q is a protocol in this $\mathbf{F}$-hybrid environment and that P is a protocol in the regular model. Then Q^P is the protocol defined as follows:

- Each message to the dummy party for $\mathbf{F}$ with $ID = (sid, pid)$ is replaced with an input to ITM running P with $ID = (sid, pid)$.
- Each output from the new party with $ID = (sid, pid)$ is treated as coming from the dummy party for $\mathbf{F}$ with the same ID.

Note that in Q^P there may be many copies of P running concurrently, but with different values of *sid*. Also note that if P is a real-life model then so is Q^P. Likewise, if P is a protocol in some $\mathbf{F}'$-hybrid model, then so is Q^P.

5 The UC Theorem

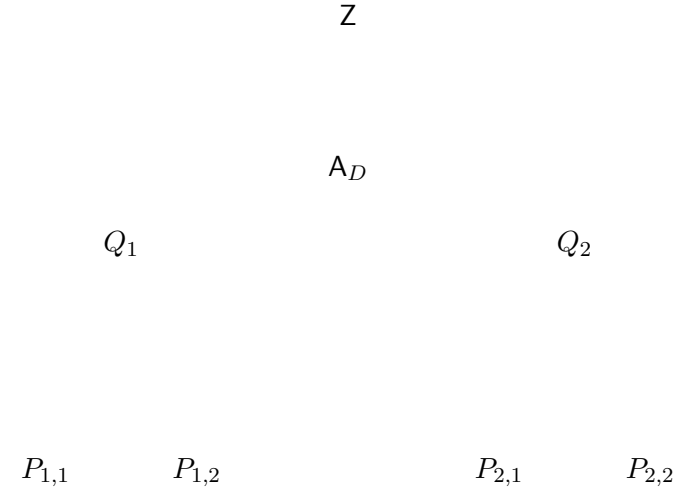
Theorem 2 *Let Q be a protocol in the $\mathbf{F}$-hybrid model and let P be a protocol that securely realizes $\mathbf{F}$. Then for every adversary A there exists another adversary H such that for every environment Z:*

$$\mathbf{Exec}^{\mathbf{F}}_{Q,\mathsf{H},\mathsf{Z}} \approx \mathbf{Exec}_{Q^P,\mathsf{A},\mathsf{Z}}$$

Proof. Taking advantage of our previous theorem, we prove this with respect to the dummy adversary A_D. That is, we will show that it is possible to simulate, in the $\mathbf{F}$-hybrid model, the execution of A_D and Q^P. To construct this simulator H, we will rely on the security of protocol P. Since P securely realizes $\mathbf{F}$, we know that there exists a simulator S_P such that no environment can distinguish between P/A_D in the real setting and $\mathbf{F}/\mathsf{S}_P$ in the ideal setting:

$$\mathbf{Ideal}^{\mathbf{F}}_{\mathsf{S}_P,\mathsf{Z}} \approx \mathbf{Exec}_{P,\mathsf{A}_D,\mathsf{Z}}$$

We will use S_P to construct the adversary H as follows:



(Single lines are for I/O, double lines for subroutine.)

Figure 3: Example of the Q^P system

- The simulator H simulates both A_D and several copies of $\mathbf{F}$.
- Messages between the machines executing protocol Q and the adversary are sent to the Q machines and the simulated A_D. (That is, H behaves exactly like A_D with regard to the machines running Q.)
- Messages that the simulated A_D would like to send to the machines running protocol P, on the other hand, are sent to the simulated S_P. Such messages must specify the *sid* and *pid* of the intended recipient. Hence, H simulates a copy of S_P for each *sid* which is used, and routes the messages accordingly. Recall that since A_D is the dummy adversary, such messages actually originate with the environment Z.
- Likewise, messages from the simulated copies of S_P are routed by H to the simulated A_D (and thus send by A_D to the adversary).
- Since there is a bijection from copies of $\mathbf{F}$ in the $\mathbf{F}$-hybrid model to executions of P in the real model, there will be a similar bijection from copies of $\mathbf{F}$ to instances of S_P. Thus, when a copy of S_P wishes to send a message to its copy of $\mathbf{F}$ (which it presumes to be the only one in the world) H routes it to the appropriate copy among the many copies of $\mathbf{F}$ available. It is essential for this that the *sid* identifiers given to the various executions of P or dummy parties for the $\mathbf{F}$ copies be unique and not conflict.

Pictorially, an example real setting (with two parties in Q, two parties in P, and two executions of P running) can be seen in Figure 3. Then the simulator is the composite machine containing the dummy adversary and the simulators as shown in Figure 4.

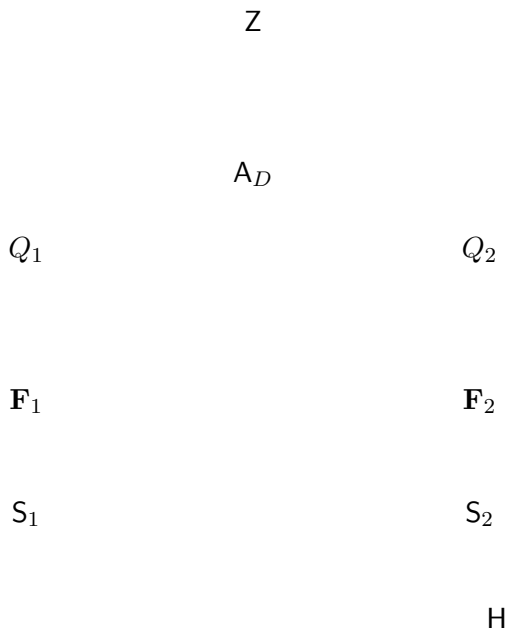


Figure 4: Example of the Q-$\mathbf{F}$ system

Now, we wish to show that this adversary H, interacting with Q in the $\mathbf{F}$-hybrid model, is indistinguishable from the dummy adversary A_D and Q^P in the real model. We return to the standard tools of computational cryptography, and show this with the hybrid argument.

Suppose that there is an environment that distinguishes the two settings. That is, there exists an environment Z so that

$$\mathbf{Ideal}^{\mathbf{F}}_{Q,\mathsf{H},\mathsf{Z}} \not\approx \mathbf{Exec}_{Q^P,\mathsf{A}_D,\mathsf{Z}}$$

We use the hybrid argument to construct an environment Z' that distinguishes a single run of P from S_P. Let m be a number which bounds the number of times P is executed in an execution of Q^P and A_D. Because the system is polynomial-time (see previous lectures) we know that m must be at most polynomial in the security parameter.Then we construct m "hybrid" systems by, in instance i, letting the first i executions of P be replaced by the simulator S_P and the functionality $\mathbf{F}$. The rest of the execution of Π remains unchanged. For the example of Figures 3 and 4, "hybrid" 1 would be as shown in Figure 5.

Thus, if Z can distinguish between "hybrid" 0 (the execution of Q^P and A_D in the real setting) and "hybrid" m (the execution of Q and H in the $\mathbf{F}$-hybrid setting) with some advantage e, then there exists an environment Z' that can distinguish between "hybrid" i and "hybrid" $i+1$ with advantage at least e/m. Thus, we can create an environment Z'' that operates as follows:

- It begins by simulating Z', A_D, and the parties running Q.
- For the first i executions of P started in the system, it simulates the functionality $\mathbf{F}$ and the simulator S_P.

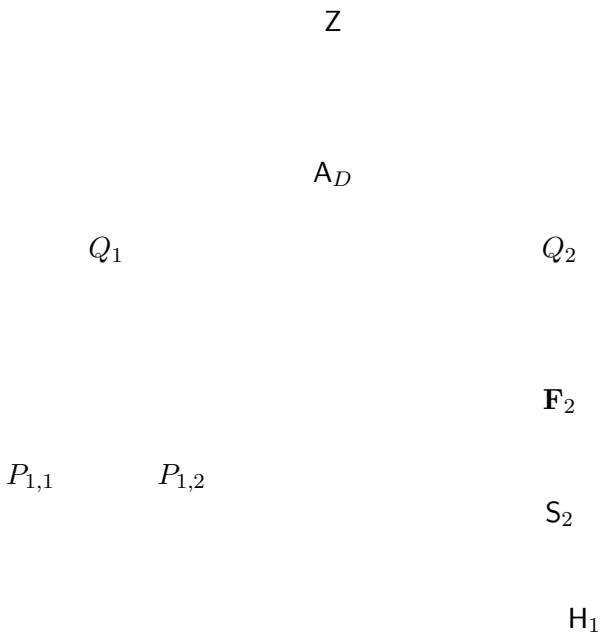


Figure 5: Example of a hybrid system

- For the $i+1$ execution of P, it starts up external real parties to execute it. Messages from the simulated A_D to parties running the $i+1$st execution of P are routed to the external dummy adversary. It also routes messages from the simulated parties to the $i+1$th execution of P to the external execution.
- For the $i+2$nd execution of P and so on, it simulates the parties to execute P.
- The output of Z'' will be exactly that of the simulated Z'.

If the external protocol is actually executed by the parties and the dummy adversary A, then the simulated Z' sees exactly the ith "hybrid" model. If the external protocol is simulated by the simulator S_P and functionality $\mathbf{F}$, then the simulated Z' sees exactly the $i+1$st "hybrid" model. Hence, A'' will distinguish the protocol P from simulation S_P with advantage at least e/m, contradicting the assumed security of P.

□

6 Implications of the UC Theorem

Corollary 1 *Let Q be a protocol that securely realizes $\mathbf{F}'$ in the $\mathbf{F}$-hybrid model, and P a protocol that securely realizes $\mathbf{F}$. Then Q^P securely realizes $\mathbf{F}'$.*

Proof. Let A be an adversary that operates against Q^P. By the composition theorem, for every adversary A there is another adversary H so that

$$\mathbf{Exec}^{\mathbf{F}}_{Q,\mathsf{H},\mathsf{Z}} \approx \mathbf{Exec}_{Q^P,\mathsf{A},\mathsf{Z}}$$

for all environments Z. Since Q securely realizes $\mathbf{F}'$ in the $\mathbf{F}$-hybrid model, there must be a simulator S' so that

$$\mathbf{Ideal}^{\mathbf{F}'}_{\mathsf{S},\mathsf{Z}} \approx \mathbf{Exec}^{\mathbf{F}}_{\mathsf{H},\mathsf{Z}}$$

Hence for every adversary A there exists a simulator S such that for every environment:

$$\mathbf{Ideal}^{\mathbf{F}'}_{\mathsf{S},\mathsf{Z}} \approx \mathbf{Exec}_{Q^P,\mathsf{A},\mathsf{Z}}$$

Thus, Q^P securely realizes $\mathbf{F}'$. □

Another implication of the theorem is that we can immediately deduce security in a multi-execution environment. That is, suppose that Q securely realizes a functionality $\mathbf{F}$. Then interacting with multiple executions of Q can be simulated using multiple instances of $\mathbf{F}$.

A last implication of the UC theorem is that protocols can be designed in a modular way. To design a protocol that securely realizes $\mathbf{F}$:

- Break $\mathbf{F}$ into sub-tasks $\mathbf{F}_1$, $\mathbf{F}_2$,... $\mathbf{F}_n$.
- Design protocols Π_i that securely realize each $\mathbf{F}_i$.
- Design a protocol Π that securely realizes $\mathbf{F}$ assuming ideal access to the $\mathbf{F}_i$ functionalities.
- Compose the protocols Π_i with protocol Π to achieve a protocol that securely realizes $\mathbf{F}$ from scratch.[1]

[1]The slides from class seem to have omitted the steps:

– Realize that your definitions of $\mathbf{F}_i$ are horribly, horribly broken.

– Fix. Lather, rinse, repeat.

– *Finally, realize what task $\mathbf{F}_i$ is all about, and redefine it entirely in light of your new intuition –Ran*