Introduction to Secure Computation, part 1

Jonathan Katz

Outline: Hour 1

- Definitions and basic feasibility results
- Composition
- Oblivious transfer (OT)

Outline: Hour 2

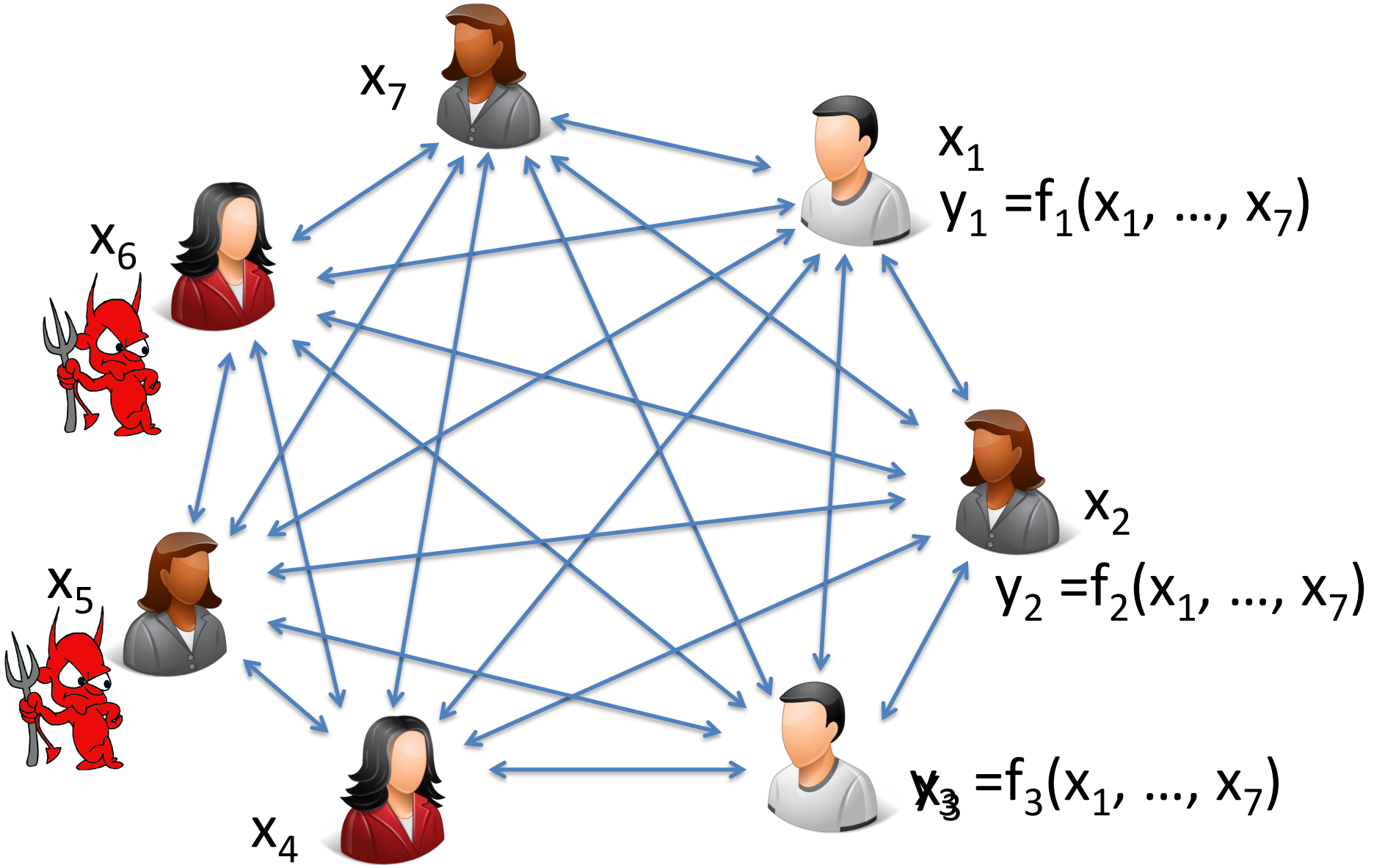
- *Semi-honest* secure computation
  - Garbled circuits and constant-round secure two-party computation
  - Secure multi-party computation
  - Constant-round secure multi-party computation

Outline: Hour 3

- Commitment schemes
- Zero knowledge proofs of knowledge (ZKPoK)
  - Dlog-based ZKPoKs
  - ZKPoK for NP
- Commitment protocols
- Coin tossing
- From semi-honest to malicious secure computation

Definitions

Real-world execution



Real-world execution

- Number of parties *n*
- Assumed bound on number of corrupted parties *t*
  - Corrupted parties assumed to *collude* and to be controlled by a single adversary/attacker
  - Static vs. adaptive corruptions
- Assumed behavior of the corrupted parties
  - Semi-honest
  - Malicious

Real-world execution

- Several aspects of the communication model need to be specified
  - Is a broadcast channel available?
  - Secure channels?
  - Synchronous communication?

Real-world execution

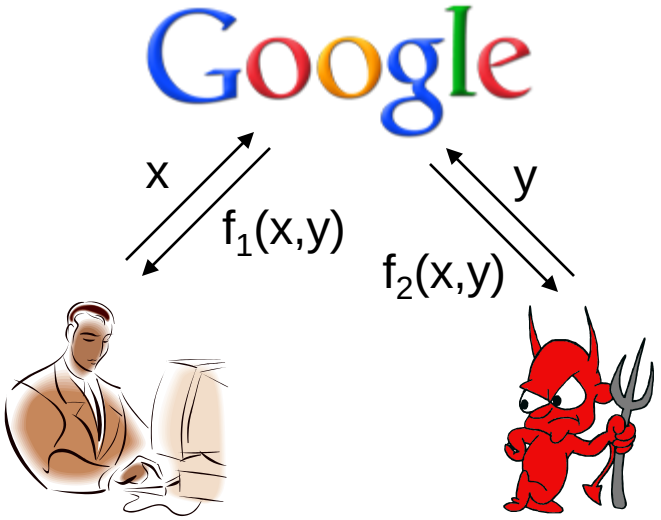
- In this talk…
  - Assume broadcast
    - Can be implemented over point-to-point channels if $t < n/3$, or if a PKI is assumed
  - Assume secure channels
    - Can be implemented with standard crypto
  - Assume synchrony
    - More difficult to justify…
    - Can consider asynchronous case also
  - Allow *rushing*
    - Adversary goes *last* in every round

Defining security?

- Could try to define security by listing several properties
  - Might depend on the function being computed
  - Might be complex to define in any case
  - How do we know we did not miss something?
- Instead…
  - Define an *ideal-world* execution
  - Compare real world to ideal world

Ideal-world execution

- Assume parties have access to a trusted party who does the computation for them



Observations

- In the ideal world:
  - *Privacy* is guaranteed (each party learns only its output)
  - *Correctness* is guaranteed (ideal party computes the correct function on inputs of the honest parties and some input from corrupted parties)
  - *Input independence* is guaranteed (corrupted parties choose inputs independently of the honest parties)
  - *True randomness* used (for randomized f's)

Ideal-world execution

- Can modify the ideal functionality to capture weaker security notions
- E.g., give up on *fairness*
  - Ideal functionality sends output to corrupted parties first
  - Attacker tells functionality to *abort* or *continue*
    - Abort: no output for honest parties
    - Continue: honest parties get correct output
  - Can consider other variations as well

Note

- Certain behaviors cannot be prevented even in the ideal world
  - Abort (no fairness)
  - Adversary can use arbitrary input(s)
  - Adversary can *infer* information about honest parties' inputs from its own output
    - See talk on differential privacy

Defining security

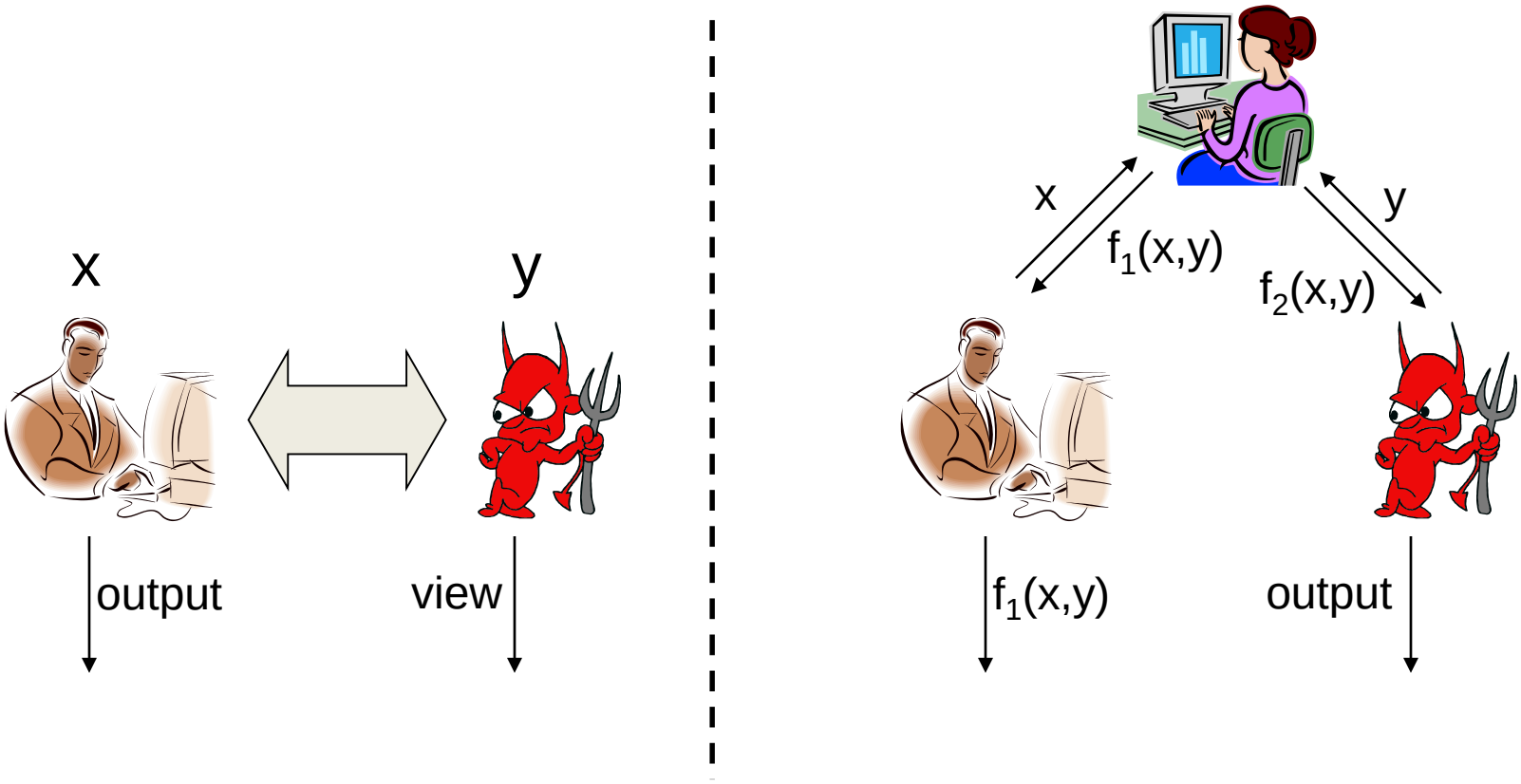
- “Real-world execution should be as secure as the ideal-world execution”
- “The only things the adversary can do in the real-world execution are things it can do in the ideal-world execution”
  - For every efficient attacker A in the real world, there is an “equivalent” (efficient) attacker B in the ideal world

Computational indistinguishability

- Distribution ensembles $A=\{A_i\}$, $B=\{B_i\}$ are *computationally indistinguishable* if for all PPT* distinguishers D
  $|\Pr[x \leftarrow A_i : D(x) = 1] - \Pr[x \leftarrow B_i : D(x) = 1]|$
  is negligible
  - Write $A \approx B$

* Can also consider non-uniform distinguishers

Defining security



A protocol is *secure* if for every (efficient) real-world adversary, there is an ideal-world adversary such that for all x, y the joint distributions of the above are computationally indistinguishable

Simulation paradigm

- To prove security, we must take an arbitrary (efficient) real-world attacker A and construct an ideal-world attacker B for which the distributions are indistinguishable for all x, y
- B is called a *simulator* for A
  - B (running in the ideal world) will *simulate* the view of A in a real-world execution
  - View = random coins + messages

Recap of security definitions

- Semi-honest security:
  - For all (efficient) *semi-honest* real-world attackers, there is an ideal-world attacker for which the real and ideal distributions are indistinguishable
- Malicious security
  - For all (efficient) real-world attackers, there is an ideal-world attacker for which the real and ideal distributions are indistinguishable
- Note ideal world is the same in both cases

Simplifications

- Suffices to consider secure computation of deterministic functions
  - Why?

- Suffices to consider secure computation where all parties get the same output
  - Why?

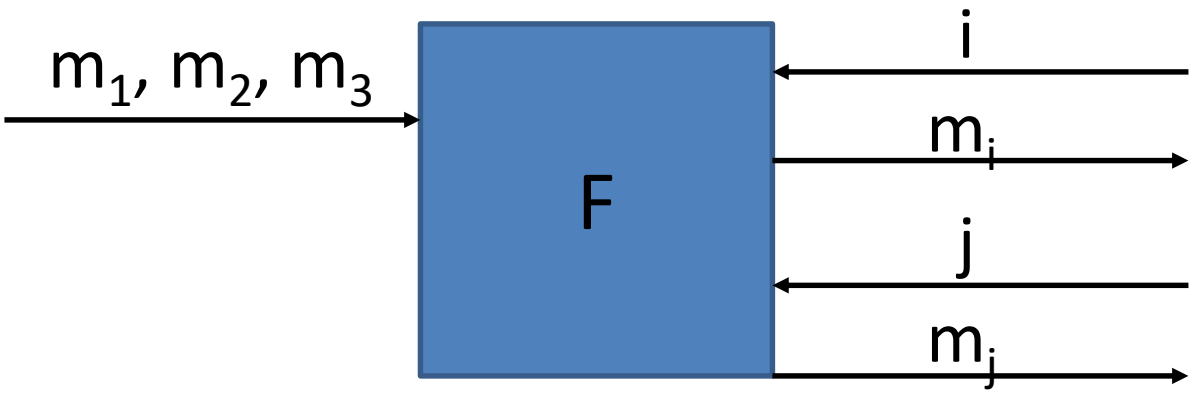
Covert security

- Intuition: cheating attacker may be successful, but is detected with high probability ε
- Modify the *ideal world* to explicitly allow the attacker to attempt cheating:
  - If corrupted party sends **cheat**
    - With probability ε, send **cheat** to honest parties; halt
    - With probability 1- ε, send honest parties' input to attacker; let attacker specify honest parties' outputs
  - Otherwise proceed as before

Reactive functionalities

- Can also consider ideal-world functionalities that maintain state (i.e., are not “one-shot”)

- E.g.:



Feasibility of secure computation?

- Under reasonable assumptions…
  secure computation of **any** function *f*, with security against **malicious** behavior of **any number** of parties, is possible
  - "Generic" result based on a *Boolean circuit* (or some other representation) for *f*
- Tradeoff between security and efficiency:
  semi-honest faster than **covert** faster than **malicious**

Composition

Composition

- Various kinds of composition to think about
  - Building a complex protocol from simpler sub-protocols
  - Running the same protocol multiple times in parallel (by the same parties)
  - Running a protocol while other protocols (run between different sets of parties) may be running

Composition

- Various kinds of composition to think about
  - **Building a complex protocol from simpler sub-protocols**
  - Running the same protocol multiple times in parallel (by the same parties)
  - Running a protocol while other protocols (run between different sets of parties) may be running

Modular (sequential) composition

- Consider computation of f in a *hybrid world* where parties run a protocol Π and **also** have access to some ideal functionalities g_1, …
  - Each g_i called sequentially
  - No protocol messages sent while g_i is called
- Can define security as before
  - Distribution in hybrid world indistinguishable from ideal-world computation of f
- What happens when we instantiate g_i with a (real-world) protocol ρ_i computing g_i?

Modular (sequential) composition

- Let $\Pi^{\rho_1, \cdots}$ denote the resulting real-world protocol
  - Sub-protocols ρ_1, … called sequentially
  - No Π-messages sent while ρ_1, … are executed
- Theorem: If $\Pi^{g_1, \cdots}$ is a secure hybrid-world protocol computing f and each ρ_i securely computes g_i, then $\Pi^{\rho_1, \cdots}$ is a secure **real-world** protocol computing f
- Very useful when designing protocols!

Modular (sequential) composition

- Caveat:
  - Some issues arise when dealing with protocols proven secure in the random-oracle model
  - Will not consider in this talk

Composition

- Various kinds of composition to think about
  - Building a complex protocol from simpler sub-protocols
  - Running the same protocol multiple times in parallel (by the same parties)
  - Running a protocol while other protocols (run between different sets of parties) may be running

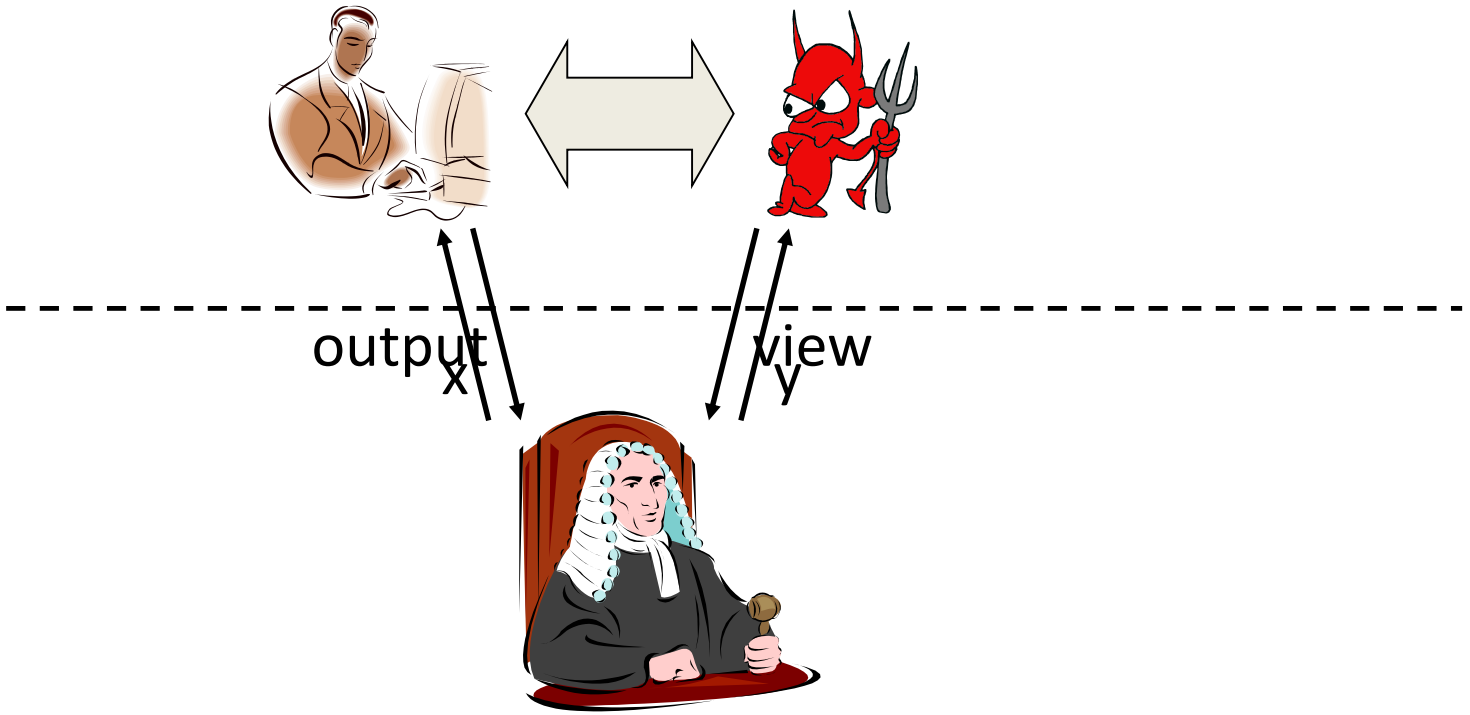
Composition

- Various kinds of composition to think about
  - Building a complex protocol from simpler sub-protocols
  - Running the same protocol multiple times in parallel (by the same parties)
  - **Running a protocol while other protocols (run between different sets of parties) may be running**

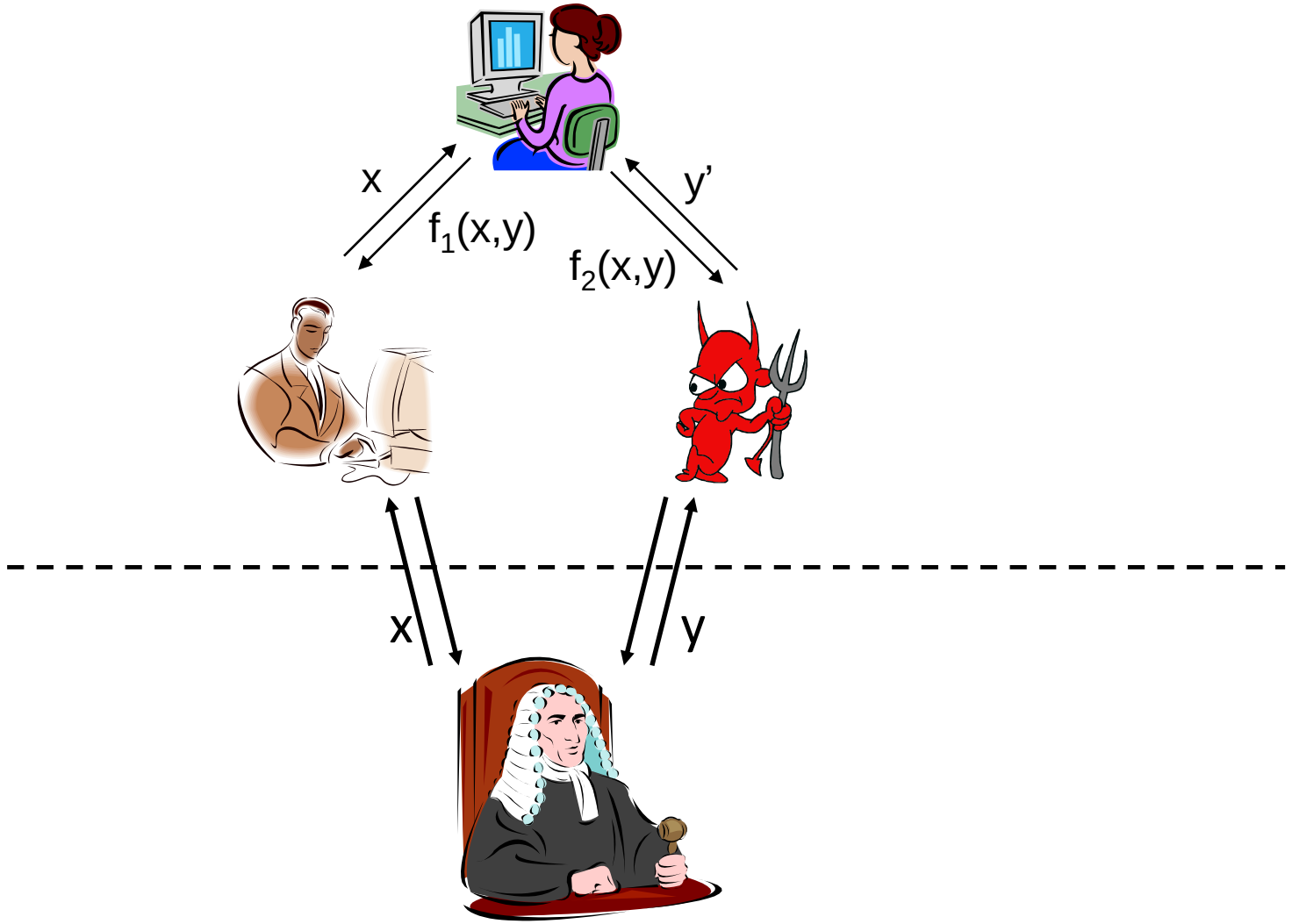
Universal composability

- (Only at a high-level…)
- So far: “stand-alone” security

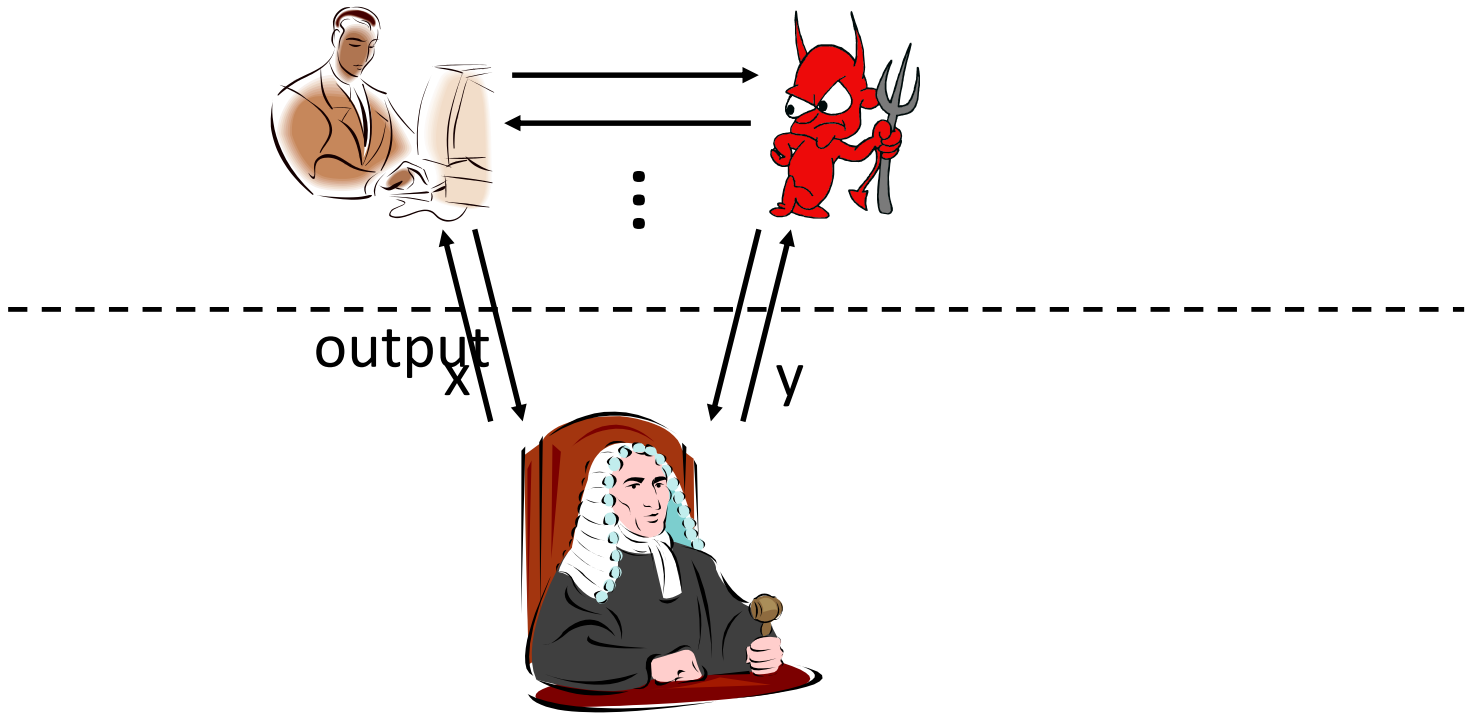
Stand-alone security



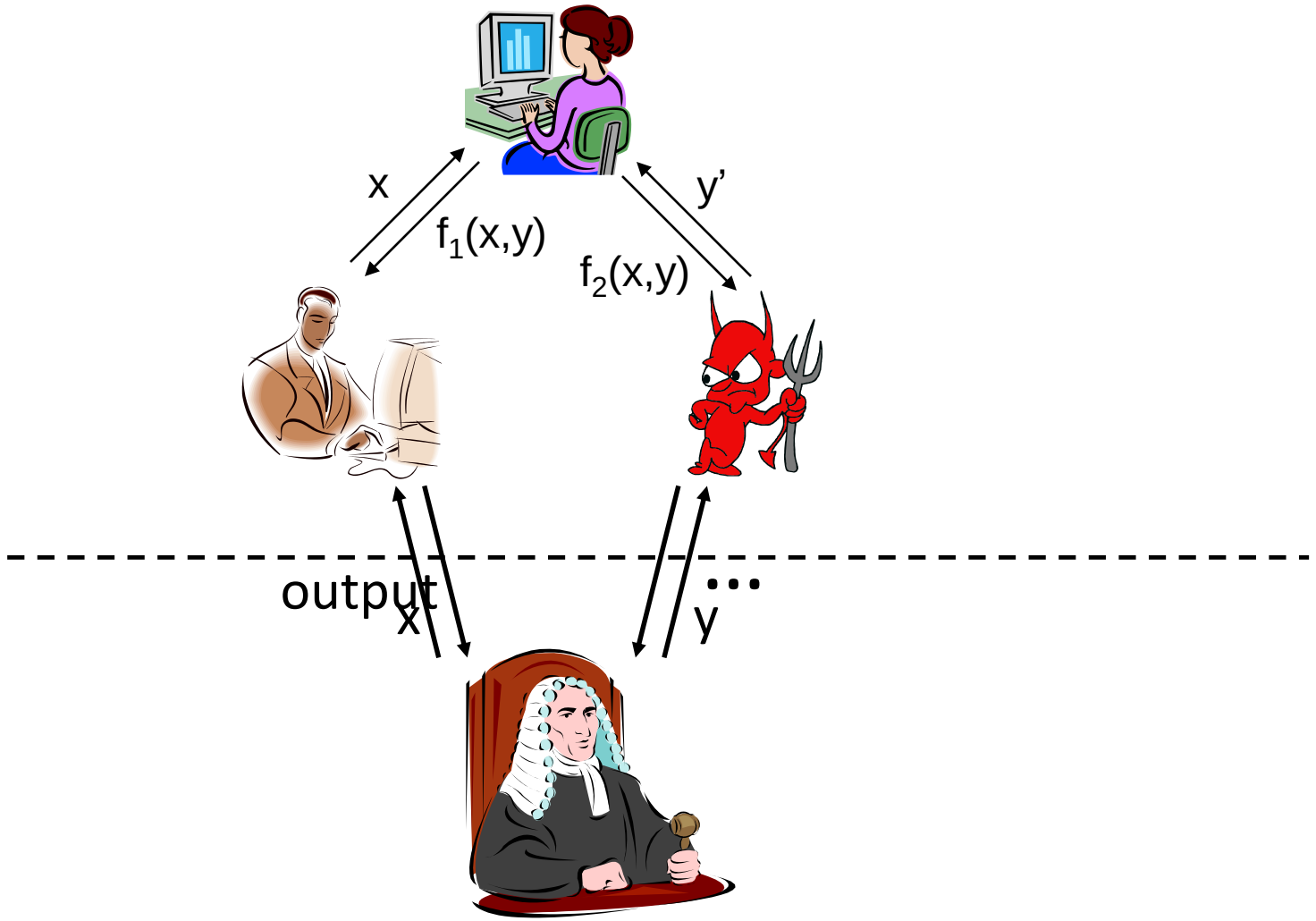
Stand-alone security



Universal composability



Universal composability



Universal composability

- A UC protocol remains secure even when executed concurrently with arbitrary other protocols

- Unfortunately, universal composability **cannot** be achieved for "most" functions without *honest majority* or *trusted setup* (CRS, RO, …)
  - We do not consider UC in the rest of the talk

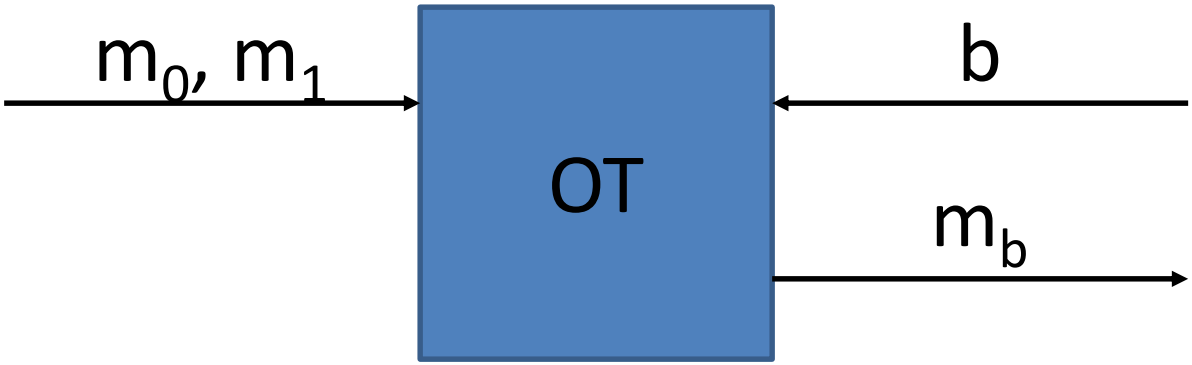
Oblivious transfer

Oblivious transfer (OT)

- Introduced by Rabin in 1982
- The workhorse of cryptographic protocols!
- Many variants…
- For now: 1-out-of-2 string OT

Oblivious transfer

Sender **Receiver**



Dlog and Diffie-Hellman assumptions

- G is a cyclic group of prime order q, with generator g

$$\log_g h = x \Leftrightarrow g^x = h$$

- Discrete-logarithm assumption: for uniform h, hard to compute $\log_g h$
- Decisional Diffie-Hellman (DDH) assumption:

$$(g, g^x, g^y, g^{xy}) \approx (g, g^x, g^y, g^z)$$

El Gamal encryption

- Parameters G, g, q
- Key generation:

  Secret key $r \leftarrow \mathbb{Z}_q$; public key is $h = g^r$
- Encryption of $m \in G$:

  $s \leftarrow \mathbb{Z}_q$; output ciphertext $(g^s, h^s \cdot m)$ denoted by $Enc_h(m)$

Semi-honest OT from DDH

Input: m_0, m_1

Input: b

$r \leftarrow \mathbb{Z}_q$

$h_b = g^r$

$h_{1-b} \leftarrow G$

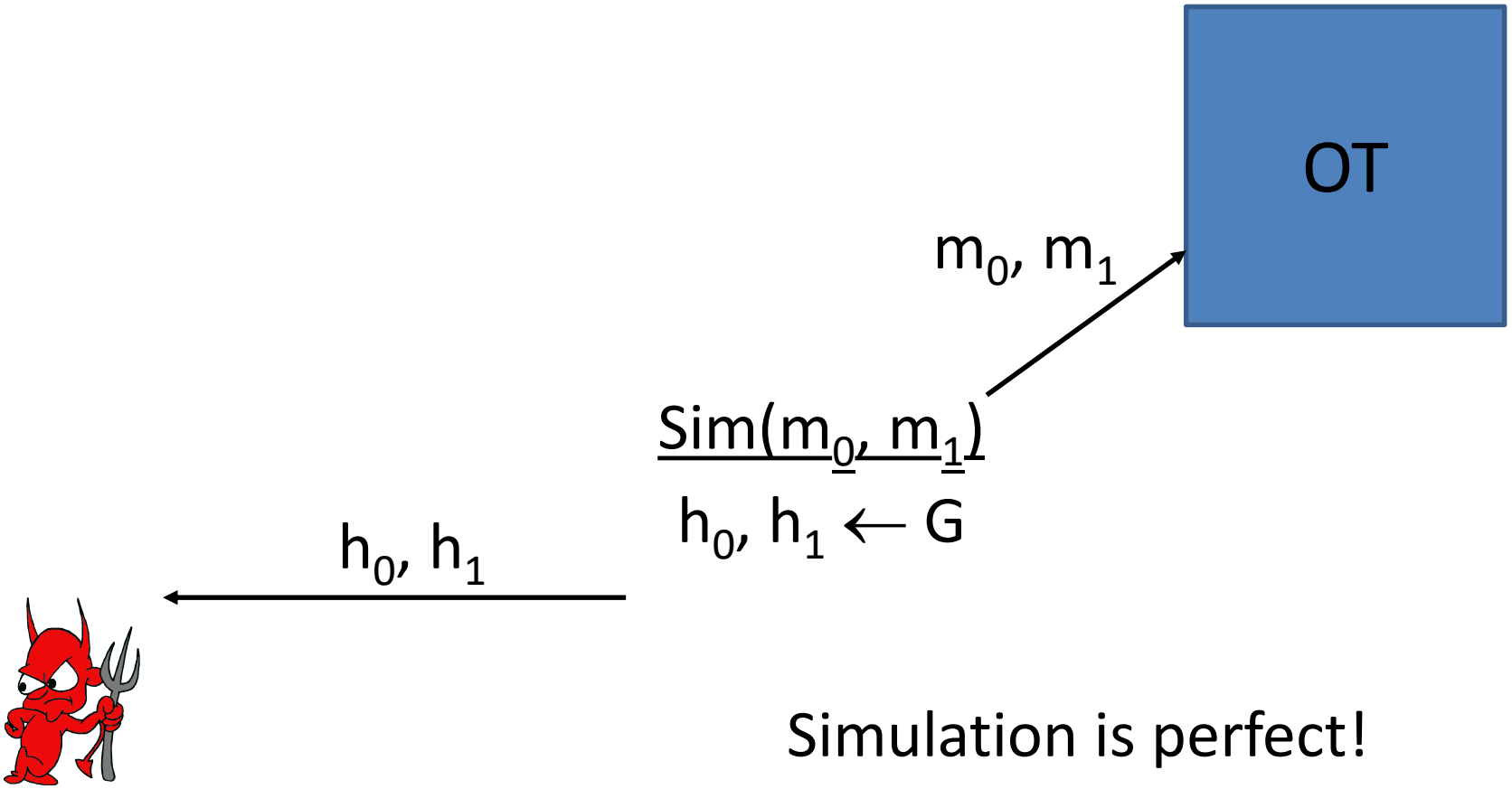
h_0, h_1 ←

$Enc_{h_0}(m_0),\ Enc_{h_1}(m_1)$ →

Decrypt…

Proof of security I

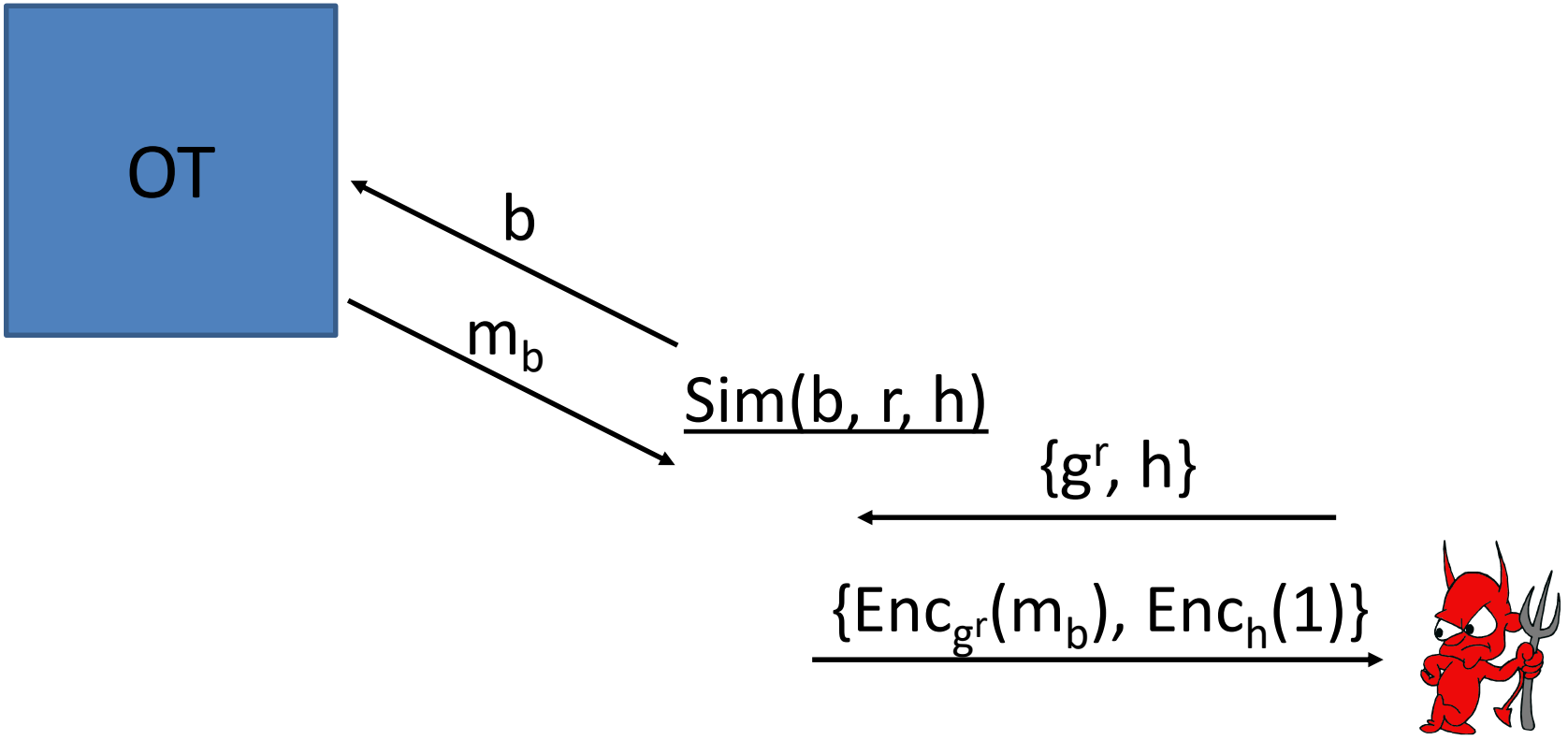
- Simulator for corrupted sender:



Simulation is perfect!

Proof of security II

- Simulator for corrupted receiver:



Indistinguishable?

Indistinguishability

- Distribution of real-world view:

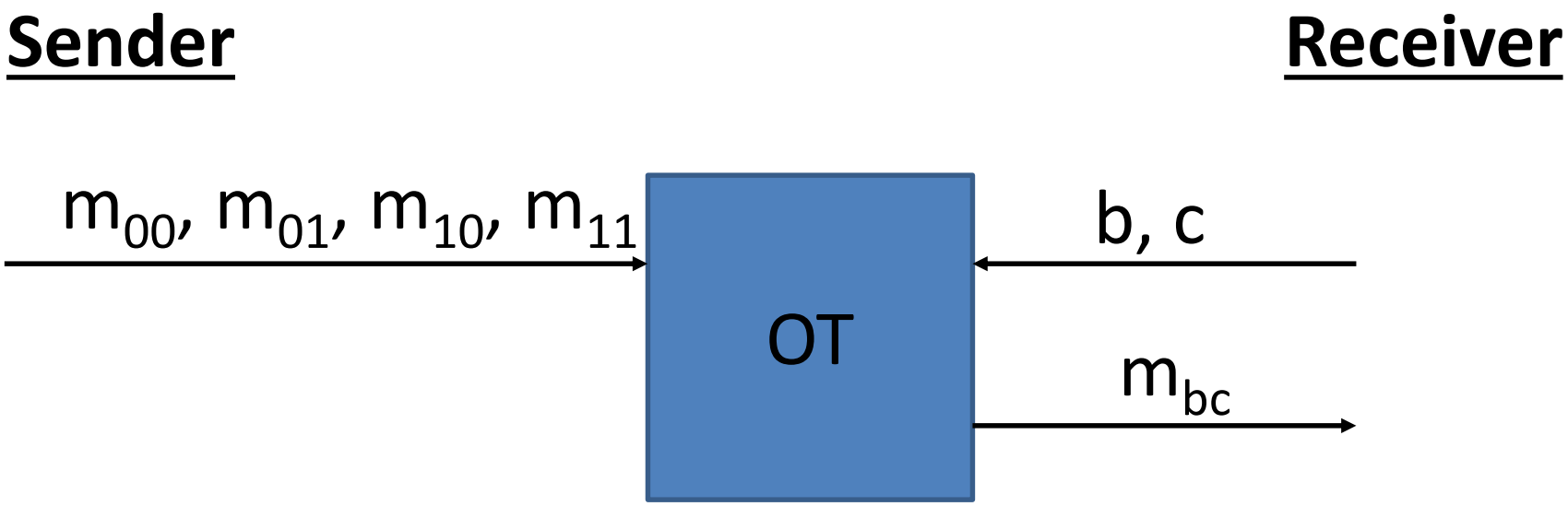
$$(r, h; \{Enc_{gr}(m_b), Enc_h(m_{1-b})\})$$

- Distribution of ideal-world view:

$$(r, h; \{Enc_{gr}(m_b), Enc_h(1)\})$$

- CPA-security of El Gamal implies that these are computationally indistinguishable!

1-of-4 oblivious transfer



Trivial to modify previous protocol to achieve this
(more-efficient solutions also possible)

Another OT protocol

Input: m_0, m_1

$h \leftarrow G$

Input: b

$r \leftarrow \mathbb{Z}_q$

$h_b = g^r$

$h_{1-b} = h/h_b$

h →

← h_0, h_1

$Enc_{h_0}(m_0), Enc_{h_1}(m_1)$ →

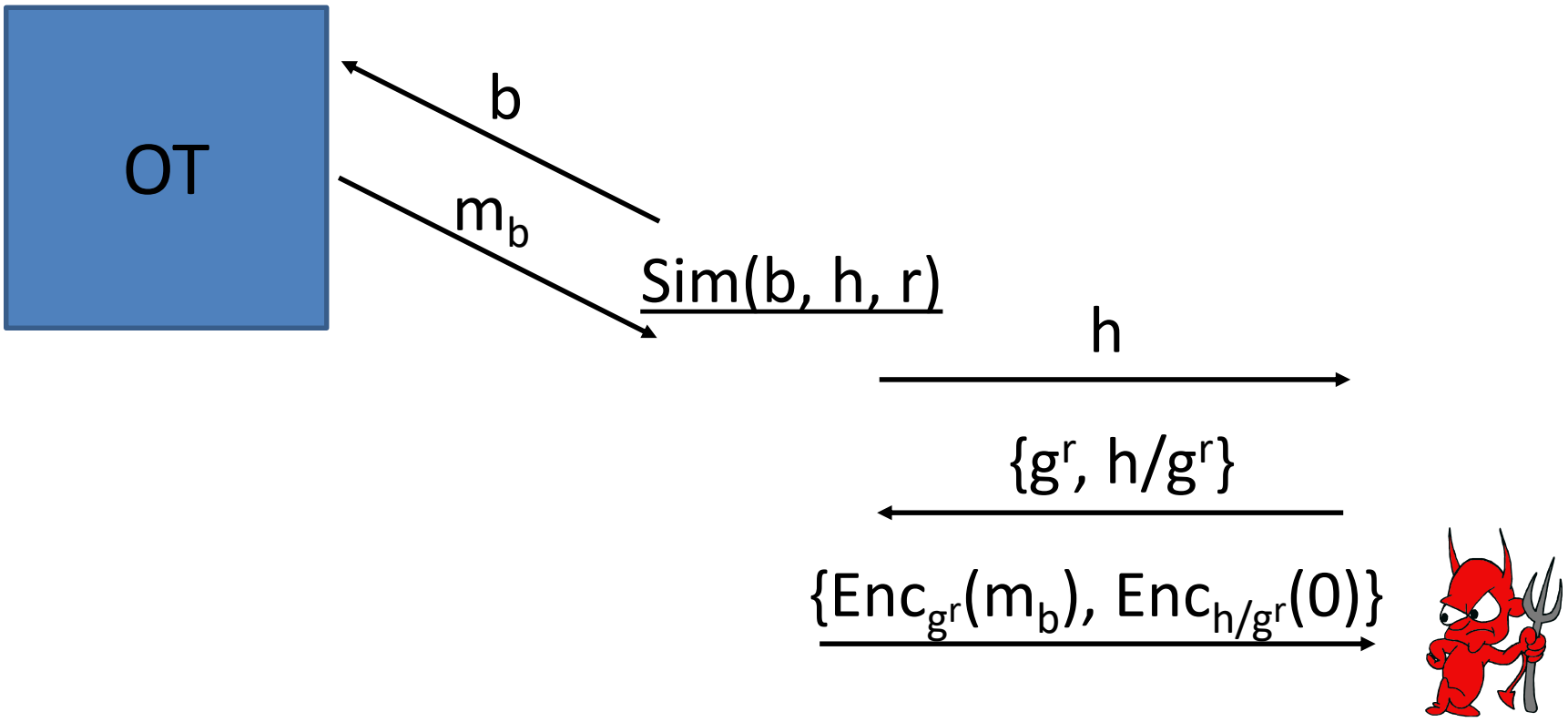
Decrypt…

Proof of security I

- Proof for corrupted sender as before
  - Simulation is perfect

Proof of security II

- Simulator for corrupted receiver:



Indistinguishable?

Indistinguishability

- Distribution of real-world view:

$$\text{REAL} = (h, r; \{\text{Enc}_{g^r}(m_b), \text{Enc}_{h/g^r}(m_{1-b})\})$$

- Distribution of ideal-world view:

$$\text{IDEAL} = (h, r; \{\text{Enc}_{g^r}(m_b), \text{Enc}_{h/g^r}(0)\})$$

- Reduction to El Gamal encryption is not immediate…

Reduction

- Use distinguisher for previous two distributions to distinguish encryption of m_{1-b} from encryption of 0
- Given (h', g_1, g_2) do
  - Choose $r \leftarrow \mathbb{Z}_q$, set $h = h' \cdot g^r$
  - Output $(h, r; \{Enc_{gr}(m_b), (g_1, g_2)\})$
- Analysis
  - If $g_1, g_2 = Enc_{h'}(m_{1-b})$, this is REAL
  - If $g_1, g_2 = Enc_{h'}(0)$, this is IDEAL

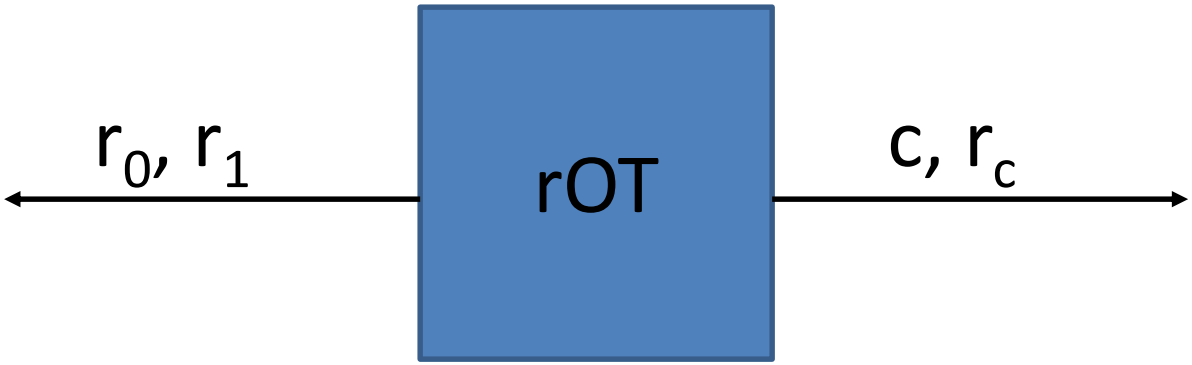
OT preprocessing

- Possible for parties to evaluate OT in advance, before their inputs are known
  - Useful for *preprocessing*

- To show result, give perfectly secure construction of OT from *random-OT*

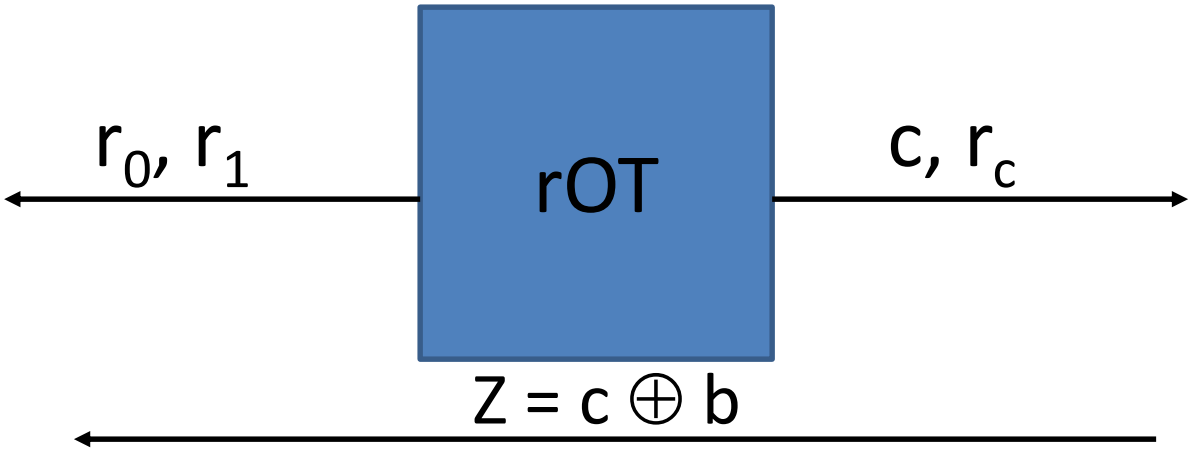
Random OT

Sender **Receiver**



OT from rOT

Sender (m_0, m_1) **Receiver (b)**



Summary

- Define security via ideal world
- Prove security using simulation paradigm
- Build larger protocols using composition
- Semi-honest OT from DDH

References

- Goldreich, “Foundations of Cryptography, vol. 2”
- Canetti, “Security and Composition of Multiparty Cryptographic Protocols,” JoC 2000
- Canetti, “Security and Composition of Cryptographic Protocols: A Tutorial,” https://eprint.iacr.org/2006/465
- Lindell, “How To Simulate It--A Tutorial on the Simulation Proof Technique,” https://eprint.iacr.org/2016/046

Thank you!