CS4501 Cryptographic Protocols
Lecture 15: OT from OT, GMW, Pseudorandomness, Garbling

https://jackdoerner.net/teaching/#2026/Spring/CS4501

Recap: 1-out-of-2 *Oblivious Transfer*

- Let $\mathscr{M}$ be some message space, and $m_0, m_1 \in \mathscr{M}$ be two messages known to P_1.
- The *Oblivious Transfer* functionality allows P_2 to receive *one* message of its choice.
- P_1 isn't allowed to learn which of the two messages P_2 received.
- P_2 isn't allowed to learn anything about the message that it didn't choose.

Recap: the π_{OT} Protocol for 1-out-of-2 OT.

- We will consider oblivious transfer for some message space $\mathscr{M} = [0,\ell]$ where ℓ is a constant. In particular, we will design a protocol that securely computes $f_{\mathsf{OT}}((m_0, m_1), b) = (\lambda, m_b)$.
- Let $\mathcal{G}$ be a PPT algorithm that takes the security parameter 1^κ and outputs $(\mathbb{G}, g, q)$, such that $q \in \mathbb{N}$ is prime, $|\mathbb{G}| = q > \ell$, $|q| \geq \kappa$, and $\langle g \rangle = \mathbb{G}$ under some operation (denoted $\cdot$).

receives inputs $1^\kappa, m_0, m_1$

$(\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa)$

$r_0 \leftarrow \mathbb{Z}_q,\ R_0 := g^{r_0},\ C_0 := X_0^{r_0} \cdot g^{m_0}$

$r_1 \leftarrow \mathbb{Z}_q,\ R_1 := g^{r_1},\ C_1 := X_1^{r_1} \cdot g^{m_1}$

$(\mathbb{G}, g, q)$ →

← (X_0, X_1)

(R_0, C_0, R_1, C_1) →

receives inputs $1^\kappa, b$

$x \leftarrow \mathbb{Z}_q,\ X_b := g^x,\ X_{1-b} \leftarrow \mathbb{G}$

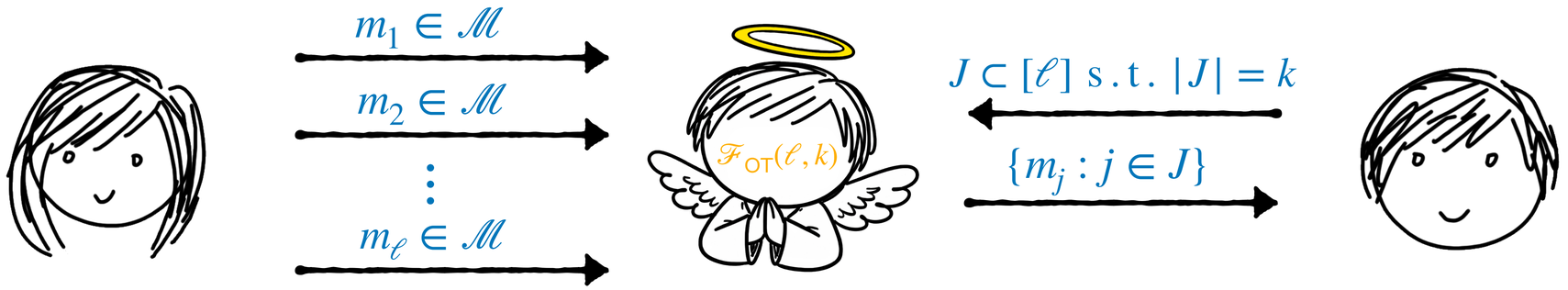
outputs $m'_b \in [0,\ell]$ s.t. $g^{m'_b} = C_b \cdot R_b^{-x}$

Theorem 1: if DDH is hard relative to $\mathcal{G}$, then the above protocol securely computes f_{OT} in the presence of a bounded semi-honest adversary that statically corrupts one party.

Proof: Homework!

Other Flavors of OT Are Available!

- It's easy to envision a generalized k-out-of-ℓ OT functionality for any $\ell > k \geq 1$.

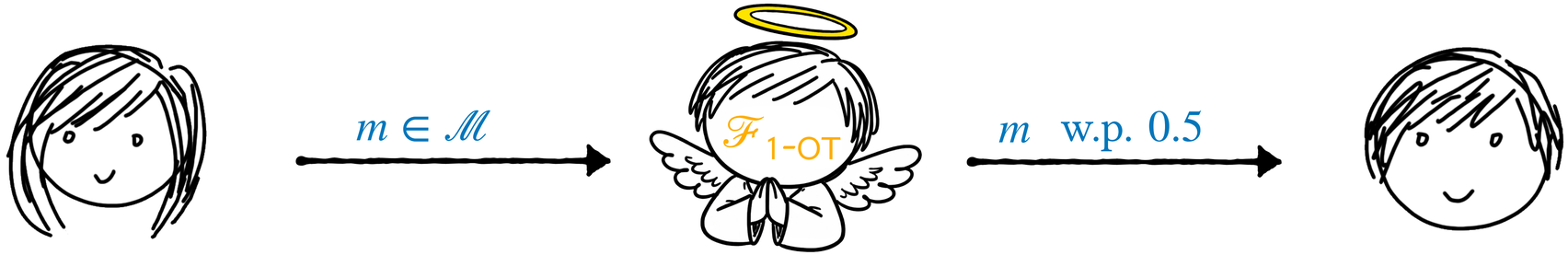


- By the end of today's class, you will see that the existence of 1-out-of-2 OT implies the existence of k-out-of-ℓ OT for every such k and ℓ.
- Intuitively, 1-out-of-2 OT implies two-party semi-honest SFE for any f; we just need to choose the right f.

Other Flavors of OT Are Available!

- 1-out-of-2 OT was introduced in 1985 by Even, Goldreich, and Lempel.

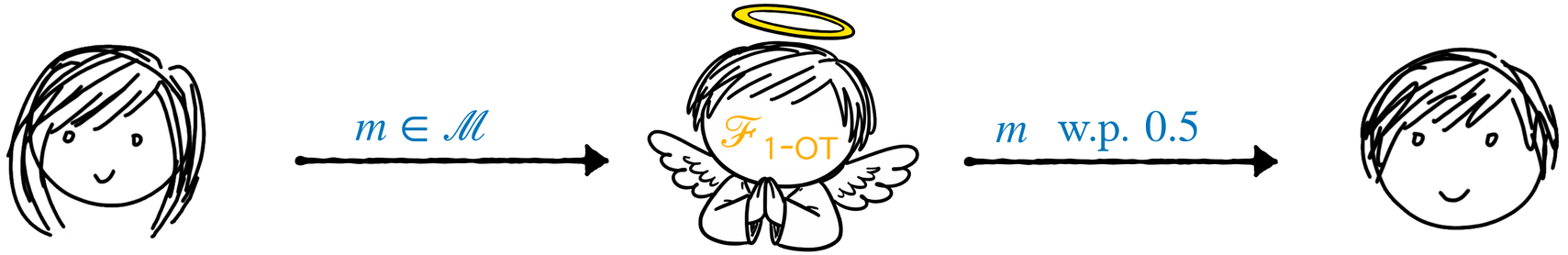
- The kind of OT originally proposed in 1981 in Michael O. Rabin's hand-written paper is a *randomized* functionality now called *1-out-of-1* OT.



- "Alice wants to send Bob a message, but her message is *randomly censored* by Captain Yossarian."

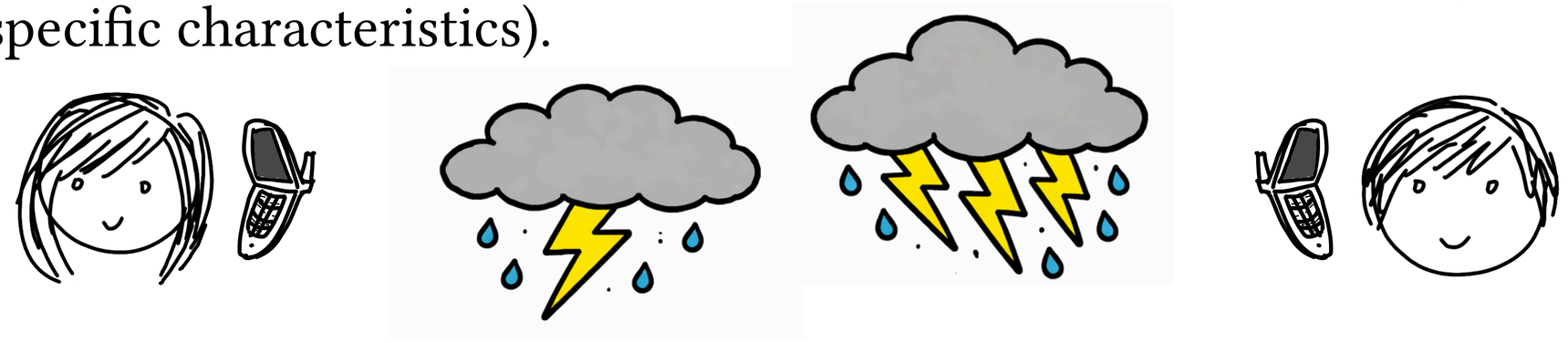
Other Flavors of OT Are Available!

- The kind of OT originally proposed in 1981 in Michael O. Rabin's hand-written paper is a *randomized* functionality now called *1-out-of-1* OT.



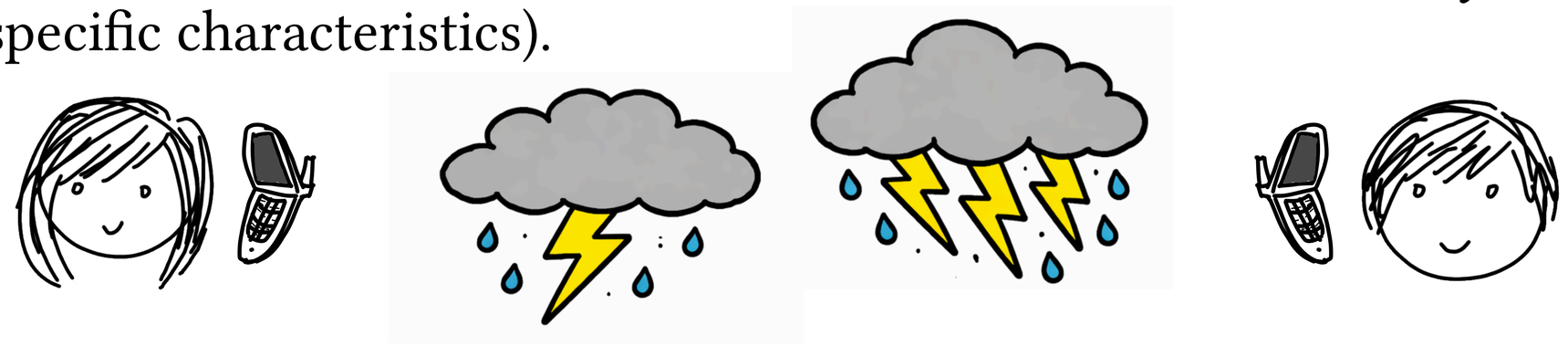
- "Alice wants to send Bob a message, but her message is *randomly censored* by Captain Yossarian."

- This models a scenario where Alice and Bob communicate over a *lossy* channel (which specific characteristics).



Some Hope if Assumptions are Wrong

- This models a scenario where Alice and Bob communicate over a *lossy* channel (which specific characteristics).



- Before moving on, we'll show that 1-out-of-1 OT implies 1-out-of-2 OT information-theoretically (i.e. without computational assumptions).
- Even if one way functions *don't exist*, there are still *physical assumptions* that we can use to build secure computation in the dishonest majority setting.
- We also know how to build OT using one way functions and *quantum communication*. This result survives even if quantum computers break all assumptions that are known to imply OT in the presence of *classical* adversaries.

1-out-of-2 OT from Lossy Channels

Theorem 2: There exists a protocol in the $\mathcal{F}_{\text{1-OT}}$-hybrid model that realizes $\mathcal{F}_{\text{OT}}$ in the presence of an unbounded semi-honest adversary who corrupts either party.

Note: We can use the definition for deterministic functionalities. We will prove that there is an *at most negligible* probability that the protocol is incorrect, and our simulator will be perfect. We get security against unbounded adversaries, which means that our security is stronger than computational. Non-zero error probability means that our security is weaker than perfect. We call this *statistical* security.

Pf Sketch:

- Assume that both functionalities use the message space $\mathcal{M} = \{0,1\}^{\ell}$.
- The Protocol (on input $(m_0, m_1) \in \mathcal{M}^2$ from Alice and $b \in \{0,1\}$ from Bob):
  1. $\forall i \in [\kappa]$, Alice samples $k_i \leftarrow \{0,1\}^{\ell}$ and sends k_i to $\mathcal{F}_{\text{1-OT}}$.

1-out-of-2 OT from Lossy Channels

- The Protocol (on input $(m_0, m_1) \in \mathcal{M}^2$ from Alice and $b \in \{0,1\}$ from Bob):

  1. $\forall i \in [\kappa]$, Alice samples $k_i \leftarrow \{0,1\}^\ell$ and sends k_i to $\mathcal{F}_{\text{1-OT}}$.

  2. Let $K_b := \{i : \text{Bob learned } k_i \text{ from } \mathcal{F}_{\text{1-OT}}\}$, and $K_{1-b} := [\kappa] \backslash K_b$.
     If $K_0 = \emptyset$ or $K_1 = \emptyset$, Bob aborts. Otherwise he sends (K_0, K_1) to Alice.

  3. For $j \in \{0,1\}$, Alice computes $c_j := m_j \oplus \hat{k}_j$ where $\hat{k}_j = \bigoplus_{i \in K_j} k_i$.

     Alice sends (c_0, c_1) to Bob.

  4. Bob computes $m'_b := c_b \oplus \hat{k}_b$ and outputs m_b.

- Correctness: Bob aborts with probability $2/2^\kappa$, which is negligible.
  If Bob does not abort, then by inspection we can see that $m'_b = m_b$.

1-out-of-2 OT from Lossy Channels

- Correctness: Bob aborts with probability $2/2^{\kappa}$, which is negligible.
  If Bob does not abort, then by inspection we can see that $m'_b = m_b$.

- Simulation for Corrupt Alice:
  For every $i \in [\kappa]$, Sim_A samples $k_i \leftarrow \{0,1\}^{\ell}$.
  For every $i \in [\kappa]$, Sim_A includes $i \in K_0$ with probability 0.5.
  Sim_A computes $K_1 := [\kappa] \backslash K_0$. Alice's view is (m_0, m_1, K_0, K_1).

  Because $K_b \equiv K_{1-b}$ in the real protocol, and $K_0 \equiv K_1$ in the output of Sim_A, this simulation is perfect.

- Simulation for Corrupt Bob:
  For every $i \in [\kappa]$, Sim_B samples $k_i \leftarrow \{0,1\}^{\ell}$ and includes it in Bob's view with probability 0.5.
  If Bob does not abort, then Sim_B also includes $c_b := m_b \oplus \hat{k}_b$ and $c_{1-b} := \hat{k}_{1-b}$.

  This distribution is also clearly identical to that of Bob's real world view. ∎

The Rest of this Lecture

Recall that we arrived at our initial definition of 1-out-of-2 OT by exploring the ways that we could simplify the BGW protocol in the case that $t = n - 1$ and the circuit is over $\mathbb{F}_2$. The protocol we arrived at is known as the GMW protocol.

1. First we will show that we can use 1-out-of-2 OT to construct 1-out-of-4 OT.

2. Then we will revisit the GMW protocol and prove its security of the 1-out-of-4-OT-hybrid model. This gives us a *feasibility theorem* for the semi-honest, dishonest majority setting.

3. In order to move our result to the plain model, we need a new composition theorem for computational security, and we need to port our perfect results to the computational setting.

4. After that, we will start to explore an *alternate* approach to secure computation that is enabled by our new computational tools.

1. 1-out-of-4 OT

1-out-of-4 OT from 1-out-of-2, for $\mathcal{M} = \{0,1\}^\ell$

receives inputs m_1, m_2, m_3, m_4

$r_1, r_2 \leftarrow \{0,1\}^\ell$

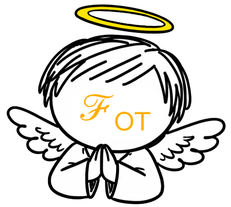
receives input $i \in [4]$

(m_1, r_1)

0 if $i = 1$, else 1

m_1 if $i = 1$, else r_1

$(m_2 \oplus r_1, r_2)$



0 if $i = 2$, else 1

$m_2 \oplus r_1$ if $i = 2$, else r_2

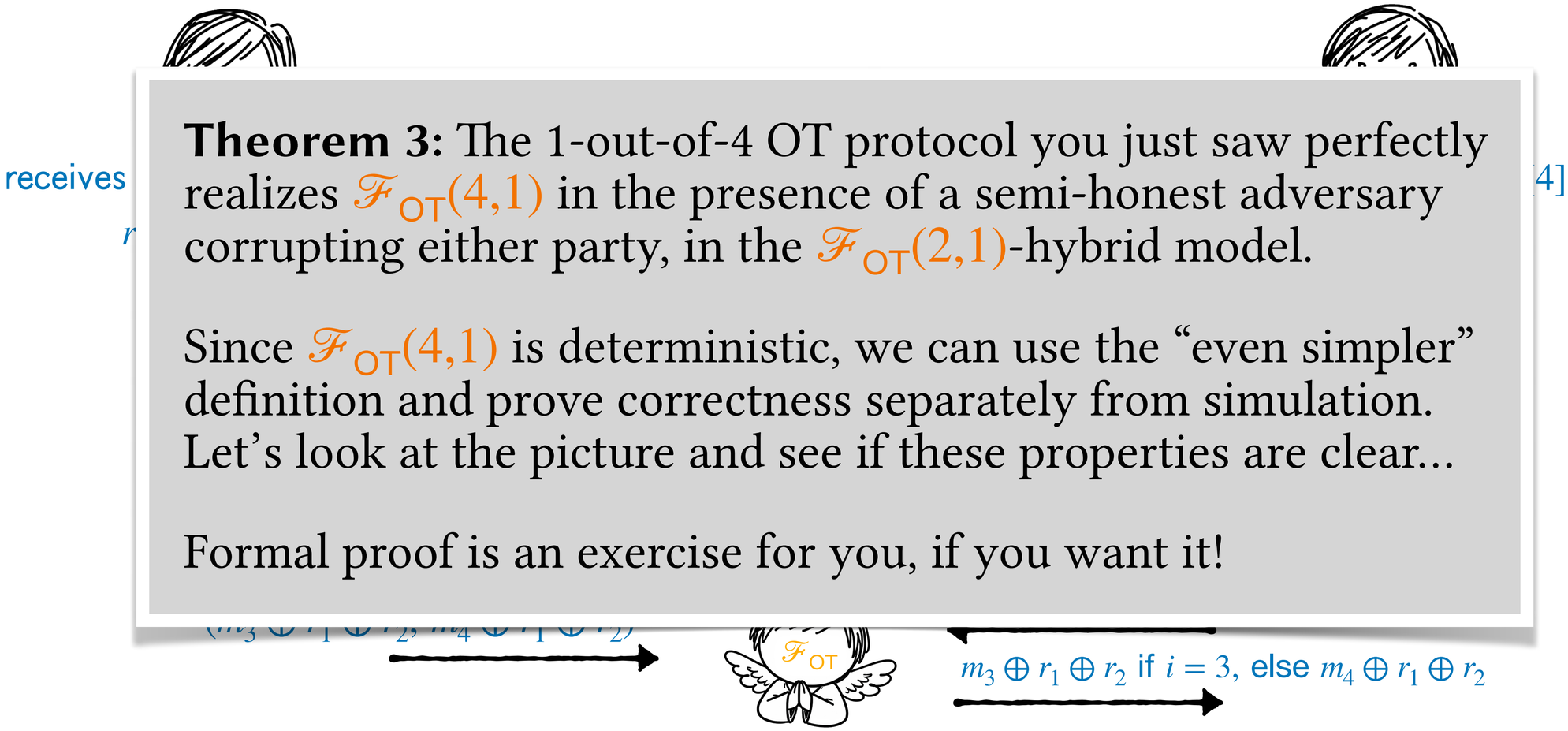
$(m_3 \oplus r_1 \oplus r_2,\ m_4 \oplus r_1 \oplus r_2)$

0 if $i = 3$, else 1

$m_3 \oplus r_1 \oplus r_2$ if $i = 3$, else $m_4 \oplus r_1 \oplus r_2$

outputs m_i

1-out-of-4 OT from 1-out-of-2, for $\mathscr{M} = \{0,1\}^\ell$



Theorem 3: The 1-out-of-4 OT protocol you just saw perfectly realizes $\mathscr{F}_{\mathsf{OT}}(4,1)$ in the presence of a semi-honest adversary corrupting either party, in the $\mathscr{F}_{\mathsf{OT}}(2,1)$-hybrid model.

Since $\mathscr{F}_{\mathsf{OT}}(4,1)$ is deterministic, we can use the "even simpler" definition and prove correctness separately from simulation. Let's look at the picture and see if these properties are clear…

Formal proof is an exercise for you, if you want it!

Is this Protocol Correct?

receives inputs m_1, m_2, m_3, m_4

$r_1, r_2 \leftarrow \{0,1\}^\ell$

receives input $i \in [4]$

(m_1, r_1)

$(m_2 \oplus r_1, r_2)$

$(m_3 \oplus r_1 \oplus r_2,\ m_4 \oplus r_1 \oplus r_2)$

0 if $i = 1$, else 1

m_1 if $i = 1$, else r_1

0 if $i = 2$, else 1

$m_2 \oplus r_1$ if $i = 2$, ~~else r_2~~

0 if $i = 3$, else 1

$m_3 \oplus r_1 \oplus r_2$ if $i = 3$, else $m_4 \oplus r_1 \oplus r_2$

If $i = 2$, then Bob learns r_1 and $m_2 \oplus r_1$

outputs m_i

Is this Protocol Correct?

receives inputs m_1, m_2, m_3, m_4

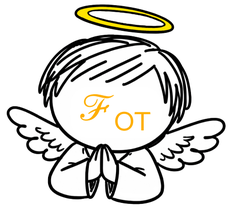
$r_1, r_2 \leftarrow \{0,1\}^\ell$

receives input $i \in [4]$

(m_1, r_1)

$(m_2 \oplus r_1, r_2)$

$(m_3 \oplus r_1 \oplus r_2,\ m_4 \oplus r_1 \oplus r_2)$



0 if $i = 1$, else 1

m_1 if $i = 1$, else r_1

0 if $i = 2$, else 1

$m_2 \oplus r_1$ if $i = 2$, else r_2

0 if $i = 3$, else 1

$m_3 \oplus r_1 \oplus r_2$ if $i = 3$, else $m_4 \oplus r_1 \oplus r_2$

Similarly if $i \in \{3,4\}$

outputs m_i

Is this Protocol Simulatable?

receives inputs m_1, m_2, m_3, m_4

$r_1, r_2 \leftarrow \{0,1\}^\ell$

receives input $i \in [4]$

Simulating Alice's view is trivial. She receives no messages and has no output.

(m_1, r_1)

0 if $i = 1$, else 1

m_1 if $i = 1$, else r_1

$(m_2 \oplus r_1, r_2)$

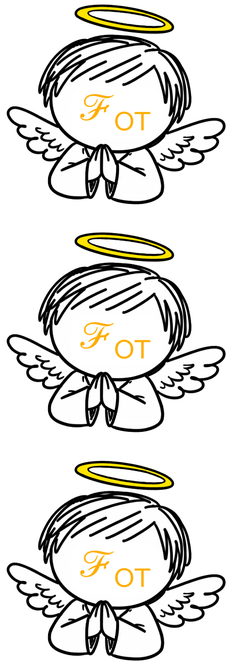
0 if $i = 2$, else 1

$m_2 \oplus r_1$ if $i = 2$, else r_2

$(m_3 \oplus r_1 \oplus r_2,\ m_4 \oplus r_1 \oplus r_2)$

0 if $i = 3$, else 1

$m_3 \oplus r_1 \oplus r_2$ if $i = 3$, else $m_4 \oplus r_1 \oplus r_2$



To simulate Bob's view, set $m_j := 0$ for all $j \neq i$ and use Bob's code.

Go through the cases one by one...

outputs m_i

Hopefully you are convinced!

2. GMW in the OT-hybrid Model

About GMW

Micali + Wigderson (and Shafi too!)

- The GMW protocol was proposed by Goldreich, Micali, and Wigderson in 1987 in the paper "How to Play any Mental Game"
- Three foundational results. We will see all of them! This is the first one.
- Since it predated BGW by one year, this was the first protocol ever written down that could securely compute *any* function.
- The very first protocol that could securely compute any function was passed down by oral tradition and not published. GMW also wrote that one down in this paper.

Simplifying Assumption: each party has exactly 1 input + output. Easy to generalize.

About GMW




Micali + Wigderson (and Shafi too!)

Simplifying Assumption: each party has exactly 1 input + output. Easy to generalize.

The basic protocol handles boolean circuits comprising the gates in, out, rand, and:

- $(\oplus, j, k, o)$ sets the value of wire o to the XOR of the values of wires j and k.
- $(\wedge, j, k, o)$ sets the value of wire o to the AND of the values of wires j and k.
- $(\neg, j, o)$ sets the value of wire o to the NOT of the values of wire j.

We will describe the protocol in the $\mathcal{F}_{\mathsf{OT}}(4,1)$-hybrid model.

The GMW Protocol $\pi_{\mathsf{GMW}}(n, t, C)$

Let $t < n \in \mathbb{N}$. There are n parties $P_1, \ldots, P_n$, with inputs $x_1, \ldots, x_n \in \mathbb{F}_2$ respectively. π_{GMW} computes a well-formed n-ary *boolean* circuit $C : \mathbb{F}_2^n \to \mathbb{F}_2^n$. The parties output $y_1, \ldots, y_n \in \mathbb{F}_2$ respectively.

There are Three Phases:

1. **Input Sharing:** every P_i with input x_i finds the entry $(\mathsf{in}, i, o) \in C$, computes $\langle w_o \rangle \leftarrow \mathsf{Share}_{2,n,t}(x_i)$ and sends $\langle w_o \rangle_j$ to every P_j for $j \in [n] \backslash \{i\}$. This takes 1 round.

2. **Circuit Eval:** the parties traverse the circuit C in topological order, jointly evaluating each gate, using shares of its input wires to produce shares of its output wire.

   - Suppose the parties arrive at gate $(\oplus, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i \oplus \langle w_k \rangle_i$. This takes 0 rounds.

The GMW Protocol $\pi_{\mathsf{GMW}}(n, t, C)$

2. **Circuit Eval:** the parties traverse the circuit C in topological order, jointly evaluating each gate, using shares of its input wires to produce shares of its output wire.

   - Suppose the parties arrive at gate $(\oplus, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i \oplus \langle w_k \rangle_i$. This takes 0 rounds.
   - Suppose the parties arrive at gate $(\wedge, j, k, o) \in C$.

     Notice that since $w_j = \bigoplus_{i \in [n]} \langle w_j \rangle_i$ and $w_k = \bigoplus_{h \in [n]} \langle w_k \rangle_h$ and $w_o = w_j \wedge w_k$,

     we need $\bigoplus_{i \in [n]} \langle w_o \rangle_i = \bigoplus_{\substack{i \in [n] \\ h \in [n]}} \langle w_j \rangle_i \wedge \langle w_k \rangle_h$.

     The parties will compute each term individually. Let's see how.

The GMW Protocol $\pi_{\mathsf{GMW}}(n, t, C)$

First, the party P_i can compute the term $\langle w_j \rangle_i \wedge \langle w_k \rangle_i$ locally.

Next, consider the pair of terms $\langle w_j \rangle_i \wedge \langle w_k \rangle_h \oplus \langle w_j \rangle_h \wedge \langle w_k \rangle_i$ for $i < h$.
Neither P_i nor P_h has enough information to compute these terms alone.

Notice that each party only has four possible input combinations.

We will take the following approach: P_i will enumerate all of the inputs that P_h could possibly have, compute the corresponding values of the above terms, and use oblivious transfer to let P_h pick (a share of) the correct one!

First, P_i samples $v_{jk}^{ih} \leftarrow \{0,1\}$.

Next, P_i prepares the following table of OT inputs:

The GMW Protocol $\pi_{\mathsf{GMW}}(n, t, C)$

Next, P_i prepares the following table of OT inputs:

Index	$\left(\langle w_k \rangle_h, \langle w_j \rangle_h\right)$	Message
1	$(0,0)$	$m_1^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$
2	$(0,1)$	$m_2^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$
3	$(1,0)$	$m_3^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$
4	$(1,1)$	$m_4^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$

P_i sends $(m_1^{ih}, m_2^{ih}, m_3^{ih}, m_4^{ih})$ to $\mathcal{F}_{\mathsf{OT}}(4,1)$, and P_h sends the index corresponding to its values of $\langle w_k \rangle_h$ and $\langle w_j \rangle_h$ per the above table.

Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\mathsf{OT}}(4,1)$.

The GMW Protocol $\pi_{\mathsf{GMW}}(n, t, C)$

Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\mathsf{OT}}(4,1)$.

Note that $v_{jk}^{ih} \oplus v_{jk}^{hi} = \langle w_j \rangle_i \wedge \langle w_k \rangle_h \oplus \langle w_j \rangle_h \wedge \langle w_k \rangle_i$.

Once every P_i and P_h for $i < h$, have computed these shares, every P_i for $i \in [n]$ finally computes:

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus\nolimits_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right).$$

This requires $n^2 - n$ invocations of $\mathcal{F}_{\mathsf{OT}}(4,1)$.

Notice that we have decomposed an n-party computation into *pairwise* invocations of a simpler functionality! This kind of pairwise decomposition will serve us well in the future!

The GMW Protocol $\pi_{\mathsf{GMW}}(n, t, C)$

- Suppose the parties arrive at gate $(\neg, j, o) \in C$. P_1 sets $\langle w_o \rangle_1 := 1 \oplus \langle w_j \rangle_1$, and every other P_i for $i > 1$ sets $\langle w_o \rangle_i := \langle w_j \rangle_i$.

- Suppose the parties arrive at gate $(\mathsf{rand}, o) \in C$. Each party P_i samples $r_{o,i} \leftarrow \{0,1\}$, and shares it to the others. The parties use a sequence of $\oplus$ gates to securely compute $\langle w_o \rangle = \bigoplus_{k \in [n]} \langle r_{o,k} \rangle$. This takes 1 round.

3. **Output Reconstruction:** Each P_i finds every output wire $(\mathsf{out}, k, j) \in C$ and sends $\langle w_j \rangle_i$ to P_k. P_k receives $\langle w_j \rangle$, computes $y_k := \mathsf{Recon}_{2,n,t}([n], \langle w_j \rangle)$, and outputs y_k. This takes 1 round.

Efficiency Note: We have defined hybrid models to exclude concurrent functionality invocations, but if we used a more advanced (and more difficult to prove) set of rules that allowed us to invoke $\mathcal{F}_{\mathsf{OT}}$ many times in parallel, the round count would be proportionate to the depth of the circuit C.

Security for the GMW Protocol

Lemma 1: Let $t < n \in \mathbb{N}$, let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be any *randomized* n-ary function, and let C be a boolean circuit that computes f. Assuming synchronicity and secure channels, $\pi_{\mathsf{GMW}}(n, t, C)$ perfectly realizes $\mathcal{F}_{\mathsf{SFE}}(n, f, \mathbb{F}_2, \ldots, \mathbb{F}_2)$ in the presence of a semi-honest $\mathcal{A}$ that statically corrupts up to t parties in the $\mathcal{F}_{\mathsf{OT}}(4,1)$-hybrid model.

Pf Sketch: We begin by enumerating the view of the corrupt parties, and then we will define the algorithm Sim. Afterward we will prove that for every $I \subseteq [n]$ of size $|I| \leq t$ and every input vector $\vec{x} \in \mathrm{domain}(f)$ it holds that

$$\left(\mathsf{Sim}\left(I, \vec{x}_I, f_I(\vec{x})\right), f(\vec{x})\right) \equiv (\mathsf{VIEW}_I, \mathsf{OUTPUT}_{\pi_{\mathsf{GMW}}})$$

See the general guidelines for semi-honest simulators from Lecture 8 for a refresher on the intuition behind the construction of the simulator algorithm.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{GMW}}(n, t, C)$ and $\mathscr{A}$

Correspondingly:
$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ is essentially a simplified version of the BGW simulator, except where AND and NOT gates are concerned.

Hybrid-World VIEW_I

$\pi_{\mathsf{GMW}}(n, t, C)$

Remember:
Apart from the AND and NOT gates, $\pi_{\mathsf{GMW}}(n, t, C)$ is essentially a simplified version of $\pi_{\mathsf{BGW}}(n, t, p, C)$ with $p = 2$.

Simulated View

$\mathsf{Sim}(I, \vec{x}_I, \vec{y}_I)$ for $\pi_{\mathsf{GMW}}(n, t, C)$ and $\mathscr{A}$

1. Copy the inputs of the corrupt parties into their views.
2. For every $h \in [n] \backslash I$, find $(\mathsf{in}, h, j) \in C$. For every $i \in I$ sample $\langle w_j \rangle_i \leftarrow \mathbb{F}_2$ and copy this value into the view of corrupted P_i.
3. For every $i \in I$, find $(\mathsf{in}, i, j) \in C$. Sample $\langle w_j \rangle \leftarrow \mathsf{Share}_{2,n,t}(x_i)$. Copy the entirety of $\langle w_j \rangle$ into the view of P_i, and for every $c \in I \backslash \{i\}$, copy $\langle w_j \rangle_c$ into the view of P_c.
4. For every $(\oplus, j, k, o) \in C$ and every $i \in I$, compute $\langle w_o \rangle_i := \langle w_j \rangle_i \oplus \langle w_k \rangle_i$ and copy this value into the view of corrupted P_i.

Hybrid-World VIEW_I

$\pi_{\mathsf{GMW}}(n, t, C)$

1. **Input Sharing:** every P_i with input x_i finds the entry $(\mathsf{in}, i, o) \in C$, computes $\langle w_o \rangle \leftarrow \mathsf{Share}_{2,n,t}(x_i)$ and sends $\langle w_o \rangle_j$ to every P_j for $j \in [n] \backslash \{i\}$.
2. **Circuit Eval:** the parties traverse the circuit C in topological order, jointly evaluating each gate, using shares of its input wires to produce shares of its output wire.
   - If the parties arrive at gate $(\oplus, j, k, o) \in C$. Each party P_i individually computes $\langle w_o \rangle_i := \langle w_j \rangle_i \oplus \langle w_k \rangle_i$.

Simulated View

Hybrid-World VIEW_I

- If the parties arrive at gate $(\wedge, j, k, o) \in C$. Each party P_i samples $v_{jk}^{ih} \leftarrow \mathbb{F}_2$ and computes

$$m_1^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$$

$$m_2^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$$

$$m_3^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$$

$$m_4^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$$

  for $h > i$. P_i sends $(m_1^{ih}, m_2^{ih}, m_3^{ih}, m_4^{ih})$ to $\mathcal{F}_{\mathsf{OT}}(4,1)$, and P_h sends the index corresponding to its values of $\langle w_k \rangle_h$ and $\langle w_j \rangle_h$. Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\mathsf{OT}}(4,1)$.

Simulated View

5. For every $(\wedge, j, k, o) \in C$ and every $i, h \in I$ such that $i < h$, sample $v_{jk}^{ih} \leftarrow \mathbb{F}_2$ and copy this value into the view of corrupted P_i.

   This is trivially identically distributed to the corresponding value in the real protocol.

Hybrid-World VIEW_I

- If the parties arrive at gate $(\wedge, j, k, o) \in C$. Each party P_i samples $v_{jk}^{ih} \leftarrow \mathbb{F}_2$ and computes

  $$m_1^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$$

  $$m_2^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$$

  $$m_3^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$$

  $$m_4^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$$

  for $h > i$. P_i sends $(m_1^{ih}, m_2^{ih}, m_3^{ih}, m_4^{ih})$ to $\mathcal{F}_{\text{OT}}(4,1)$, and P_h sends the index corresponding to its values of $\langle w_k \rangle_h$ and $\langle w_j \rangle_h$. Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\text{OT}}(4,1)$.

  Notice that if P_i is corrupt, the only new value in its view is uniformly sampled.

Simulated View

5. For every $(\wedge, j, k, o) \in C$ and every $i, h \in I$ such that $i < h$, sample $v_{jk}^{ih} \leftarrow \mathbb{F}_2$ and copy this value into the view of corrupted P_i.

   This is trivially identically distributed to the corresponding value in the real protocol.

6. For every $(\wedge, j, k, o) \in C$ and every $i, h \in I$ such that $i < h$, compute $v_{jk}^{hi} := \langle w_j \rangle_i \wedge \langle w_k \rangle_h \oplus \langle w_j \rangle_h \wedge \langle w_k \rangle_i \oplus v_{jk}^{ih}$ and copy this value into the view of corrupted P_h.

   This is identically distributed to the corresponding value in the real protocol because it is computed in the same way, from identically distributed values.

Hybrid-World VIEW_I

- If the parties arrive at gate $(\wedge, j, k, o) \in C$. Each party P_i samples $v_{jk}^{ih} \leftarrow \mathbb{F}_2$ and computes

  $$m_1^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$$

  $$m_2^{ih} := \langle w_j \rangle_i \wedge 0 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$$

  $$m_3^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 0 \oplus v_{jk}^{ih}$$

  $$m_4^{ih} := \langle w_j \rangle_i \wedge 1 \oplus \langle w_k \rangle_i \wedge 1 \oplus v_{jk}^{ih}$$

  for $h > i$. P_i sends $(m_1^{ih}, m_2^{ih}, m_3^{ih}, m_4^{ih})$ to $\mathcal{F}_{\text{OT}}(4,1)$, and P_h sends the index corresponding to its values of $\langle w_k \rangle_h$ and $\langle w_j \rangle_h$. Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\text{OT}}(4,1)$.

  Notice that if P_i is corrupt, the only new value in its view is uniformly sampled.

Simulated View

Hybrid-World VIEW_I

for $h > i$. P_i sends $(m_1^{ih}, m_2^{ih}, m_3^{ih}, m_4^{ih})$ to $\mathcal{F}_{\mathsf{OT}}(4,1)$, and P_h sends the index corresponding to its values of $\langle w_k \rangle_h$ and $\langle w_j \rangle_h$. Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\mathsf{OT}}(4,1)$.

Notice that if P_i is corrupt, the only new value in its view is uniformly sampled.

If P_h is corrupt but P_i is honest, the only new value in its view is *masked* by the uniformly sampled v_{jk}^{ih} that it does not know.

Simulated View

7. For every $(\wedge, j, k, o) \in C$ and every $h \in I$ and $i \in [n] \backslash I$ such that $i < h$, sample $v_{jk}^{hi} \leftarrow \mathbb{F}_2$ and copy this value into the view of corrupted P_i.

   See note at left. Identical distribution for this value follows.

Hybrid-World VIEW_I

for $h > i$. P_i sends $(m_1^{ih}, m_2^{ih}, m_3^{ih}, m_4^{ih})$ to $\mathcal{F}_{\text{OT}}(4,1)$, and P_h sends the index corresponding to its values of $\langle w_k \rangle_h$ and $\langle w_j \rangle_h$. Let $v_{jk}^{hi} \in \{0,1\}$ denote value that P_h receives from $\mathcal{F}_{\text{OT}}(4,1)$.

Notice that if P_i is corrupt, the only new value in its view is uniformly sampled.

If P_h is corrupt but P_i is honest, the only new value in its view is *masked* by the uniformly sampled v_{jk}^{ih} that it does not know.

Simulated View

Hybrid-World VIEW_I

Once every P_i and P_h for $i < h$, have computed these shares, every P_i for $i \in [n]$ finally computes:

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus\nolimits_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right).$$

Simulated View

8. For every $(\wedge, j, k, o) \in C$ and every $i \in I$, compute

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right)$$

and copy this value into the view of corrupted P_i.

This is identically distributed to the corresponding value in the real protocol because it is computed in the same way, from identically distributed values.

Hybrid-World VIEW_I

Once every P_i and P_h for $i < h$, have computed these shares, every P_i for $i \in [n]$ finally computes:

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right).$$

Simulated View

8. For every $(\wedge, j, k, o) \in C$ and every $i \in I$, compute

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right)$$

and copy this value into the view of corrupted P_i.

This is identically distributed to the corresponding value in the real protocol because it is computed in the same way, from identically distributed values.

Hybrid-World VIEW_I

Once every P_i and P_h for $i < h$, have computed these shares, every P_i for $i \in [n]$ finally computes:

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right).$$

- If the parties arrive at gate $(\neg, j, o) \in C$. P_1 sets $\langle w_o \rangle_1 := 1 \oplus \langle w_j \rangle_1$, and every other P_i for $i > 1$ sets $\langle w_o \rangle_i := \langle w_j \rangle_i$.

Simulated View

8. For every $(\wedge, j, k, o) \in C$ and every $i \in I$, compute

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right)$$

and copy this value into the view of corrupted P_i.

This is identically distributed to the corresponding value in the real protocol because it is computed in the same way, from identically distributed values.

9. For every $(\neg, j, o) \in C$ and every $i \in I$, compute $\langle w_o \rangle_i := \langle w_j \rangle_i$ if $i > 1$, or $\langle w_o \rangle_1 := 1 \oplus \langle w_j \rangle_1$ and copy this value into the view of corrupted P_i.

Hybrid-World VIEW_I

Once every P_i and P_h for $i < h$, have computed these shares, every P_i for $i \in [n]$ finally computes:

$$\langle w_o \rangle_i := \langle w_j \rangle_i \wedge \langle w_k \rangle_i \oplus \left(\bigoplus_{h \in [n] \setminus \{i\}} v_{jk}^{ih} \right).$$

- If the parties arrive at gate $(\neg, j, o) \in C$. P_1 sets $\langle w_o \rangle_1 := 1 \oplus \langle w_j \rangle_1$, and every other P_i for $i > 1$ sets $\langle w_o \rangle_i := \langle w_j \rangle_i$.

Simulated View

10. For every $(\mathsf{rand}, o) \in C$ and $i \in I$, sample $r_{o,i} \leftarrow \mathbb{F}_2$ and $\langle r_{o,i} \rangle \leftarrow \mathsf{Share}_{2,n,t}(r_{o,i})$, and for $h \in [n] \backslash I$, sample $\langle r_{o,h} \rangle_i \leftarrow \mathbb{F}_2$. Compute $\langle w_o \rangle_i := \sum_{k \in [n]} \langle r_{o,k} \rangle_i$ and copy $r_{o,i}$, $\langle r_{o,i} \rangle$, $\langle w_o \rangle_i$ and $\langle r_{o,k} \rangle_i \; \forall k \in [n] \backslash \{i\}$ into the view of corrupted P_i.
11. For $i \in I$, find $(\mathsf{out}, i, j) \in C$. Sim received y_i as input, and computed $\langle w_j \rangle_c \; \forall c \in I$ in the previous steps. For every $h \in [n] \backslash I$, sample $\langle w_j \rangle_h \leftarrow \mathbb{F}_2$ subject to the constraint that $\bigoplus_{k \in [n]} \langle w_j \rangle_k = y_i$, and copy $\langle w_j \rangle_h$ into the view of P_i.

Hybrid-World VIEW_I

- If the parties arrive at gate $(\mathsf{rand}, o) \in C$. Each party P_i samples $r_{o,i} \leftarrow \mathbb{F}_2$, the parties use a sequence of in gates and $\oplus$ gates to securely compute $\langle w_o \rangle$ such that $w_o = \sum_{k \in [n]} r_{o,k}$.

3. **Output Reconstruction:** Each P_i finds every output wire $(\mathsf{out}, k, j) \in C$ and sends $\langle w_j \rangle_i$ to P_k. P_k receives $\langle w_j \rangle$, computes $y_k := \mathsf{Recon}_{2,n,t}([n], \langle w_j \rangle)$, and outputs y_k.

The rest of the argument is essentially identical to the proof of security for BGW in the $\mathcal{F}_{\mathsf{mul}}$-hybrid model. See Lecture 8 for a refresher! ∎

Security for the GMW Protocol

By the composition theorem for perfectly-secure protocols (Lecture 10), we get:

Corollary 1: Let $t < n \in \mathbb{N}$, let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be any *randomized* n-ary function, and let C be a boolean circuit that computes f. Assuming synchronicity and secure channels, there exists a protocol that perfectly realizes $\mathcal{F}_{\mathsf{SFE}}(n, f, \mathbb{F}_2, \ldots, \mathbb{F}_2)$ in the presence of a semi-honest $\mathcal{A}$ that statically corrupts up to t parties in the $\mathcal{F}_{\mathsf{OT}}(2,1)$-hybrid model.

Now we have a problem. In order to get plain-model feasibility for dishonest majority secure computation in the presence of a dishonest majority, we need to replace $\mathcal{F}_{\mathsf{OT}}(2,1)$ with π_{OT} from the beginning of the lecture.

However, you will be proving that π_{OT} *computationally* realizes $\mathcal{F}_{\mathsf{OT}}(2,1)$, whereas $\pi_{\mathsf{GMW}}(n, t, C)$ *perfectly* realizes $\mathcal{F}_{\mathsf{SFE}}(n, f, \mathbb{F}_2, \ldots, \mathbb{F}_2)$. We need a way to mix computational and perfectly secure components, and get a sane outcome!

Security for the GMW Protocol

Now we have a problem. In order to get plain-model feasibility for dishonest majority secure computation in the presence of a dishonest majority, we need to replace $\mathcal{F}_{\mathsf{OT}}(2,1)$ with π_{OT} from the beginning of the lecture.

However, you will be proving that π_{OT} *computationally* realizes $\mathcal{F}_{\mathsf{OT}}(2,1)$, whereas $\pi_{\mathsf{GMW}}(n,t,C)$ *perfectly* realizes $\mathcal{F}_{\mathsf{SFE}}(n,f,\mathbb{F}_2,\ldots,\mathbb{F}_2)$. We need a way to mix computational and perfectly secure components, and get a sane outcome!

Resolving this discrepancy requires two steps: first, we will show that any perfectly secure protocol is also computationally secure, with one extra restriction.

Then we will assert a composition theorem for computationally secure protocols. Proving it is out of scope for this class. Even Goldreich's textbook only addresses the two-party case in detail.

3. Perfect and Computational, Together

Perfect Protocols with Efficient Simulation

Lemma 2: Let $\mathfrak{X} = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $\mathfrak{Y} = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ be ensembles. If $\mathfrak{X} \equiv \mathfrak{Y}$, then $\mathfrak{X} \approx_c \mathfrak{Y}$.

Proof: For every $\kappa \in \mathbb{N}$, $X_\kappa \equiv Y_\kappa$, which implies that $\forall S \subseteq (\text{domain}(X_\kappa) \cup \text{domain}(Y_\kappa))$ we have $\Pr[x \in S : x \leftarrow X_\kappa] = \Pr[y \in S : y \leftarrow Y_\kappa]$.

Next, observe that any PPT distinguisher with a fixed security parameter 1^κ can be viewed as a deterministic algorithm taking a random string r from some finite set $R_{\mathscr{D},\kappa}$.

If for every PPT $\mathscr{D}$, $\kappa \in \mathbb{N}$, and $r \in R_{\mathscr{D},\kappa}$ we let $T_{\mathscr{D},\kappa,r} = \{x : \mathscr{D}(1^\kappa, x; r) = 1\}$, then we have $\Pr[x \in T_{\mathscr{D},\kappa,r} : x \leftarrow X_\kappa] = \Pr[y \in T_{\mathscr{D},\kappa,r} : y \leftarrow Y_\kappa]$, which finally implies

$$\left|\Pr[\mathscr{D}(1^\kappa, x) = 1 : x \leftarrow X_\kappa] - \Pr[\mathscr{D}(1^\kappa, y) = 1 : y \leftarrow Y_\kappa]\right| = 0. \ \blacksquare$$

Corollary 2: If a PPT protocol π perfectly realizes a functionality $\mathscr{F}$, and for every PPT $\mathscr{A}$ the corresponding simulator is also PPT, then π computationally realizes $\mathscr{F}$.

Recall: Composition Theorem

Perfect MPC Composition Theorem: Let $t < n$.

- Let π_{outer} be an n-party protocol in the $\mathcal{F}_{\text{inner}}$-hybrid model that perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- Let π_{inner} be an n-party protocol that perfectly realizes $\mathcal{F}_{\text{inner}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- If $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ is π_{outer} with every call to $\mathcal{F}_{\text{inner}}$ replaced by an invocation of π_{inner}.
- Then $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ perfectly realizes $\mathcal{F}_{\text{outer}}$ in the presence of a semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.

Now: Computational Composition Theorem

~~**Perfect**~~ **MPC Composition Theorem:** Let $t < n$.

- Let π_{outer} be an n-party protocol in the $\mathcal{F}_{\text{inner}}$-hybrid model that computationally realizes $\mathcal{F}_{\text{outer}}$ in the presence of a bounded semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- Let π_{inner} be an n-party protocol that computationally realizes $\mathcal{F}_{\text{inner}}$ in the presence of a bounded semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.
- If $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ is π_{outer} with every call to $\mathcal{F}_{\text{inner}}$ replaced by an invocation of π_{inner}.
- Then $\pi_{\text{outer}}^{\mathcal{F}_{\text{inner}} \to \pi_{\text{inner}}}$ computationally realizes $\mathcal{F}_{\text{outer}}$ in the presence of a bounded semi-honest adversary that statically corrupts up to t parties, assuming synchrony and secure point-to-point channels.

Feasibility for Dishonest Majority MPC

Note: *every* simulator we saw in class was PPT if its corresponding adversary was PPT.

Finally, putting the above fact together with Corollary 1 and Corollary 2 implies that there exists a protocol that computationally realizes $\mathcal{F}_{\mathsf{SFE}}(n, f, \mathbb{F}_2, \ldots, \mathbb{F}_2)$ in the presence of a bounded semi-honest $\mathcal{A}$ that statically corrupts up to t parties in the $\mathcal{F}_{\mathsf{OT}}(2,1)$ -hybrid model, for any $t < n$.

Putting this together with the fact that π_{OT} computationally realizes $\mathcal{F}_{\mathsf{OT}}(2,1)$ (which you will prove for homework!) yields:

Dishonest Majority MPC Feasibility Theorem: Let $t < n$. Assuming synchrony, secure point-to-point channels, and that the Decisional Diffie-Hellman Assumption is true, there exists an n-party protocol that securely computes any randomized, polynomial-time function f in the presence of a semi-honest adversary that statically corrupts up to t parties.

Syllabus (tentative):

A taxonomy of adversaries; a variety of techniques

(now the taxonomy should be clearer than it was before)

	Part 1: *Information-theoretic* techniques. Adversaries with unbounded power	**Part 2:** *Cryptographic* techniques. Adversaries with bounded power
Semi-honest Adversaries: *follow* the rules of the protocol	Secret Sharing BGW protocol for an honest majority	Oblivious Transfer GMW protocol for a dishonest majority Yao's protocol for two parties Fully Homomorphic Encryption
Malicious Adversaries: *break* the rules of the protocol	Verifiable Secret Sharing BGW protocol for honest supermajority	Coin Tossing Zero-Knowledge Proofs GMW Compiler Byzantine Agreement + Broadcast

Overarching Questions:

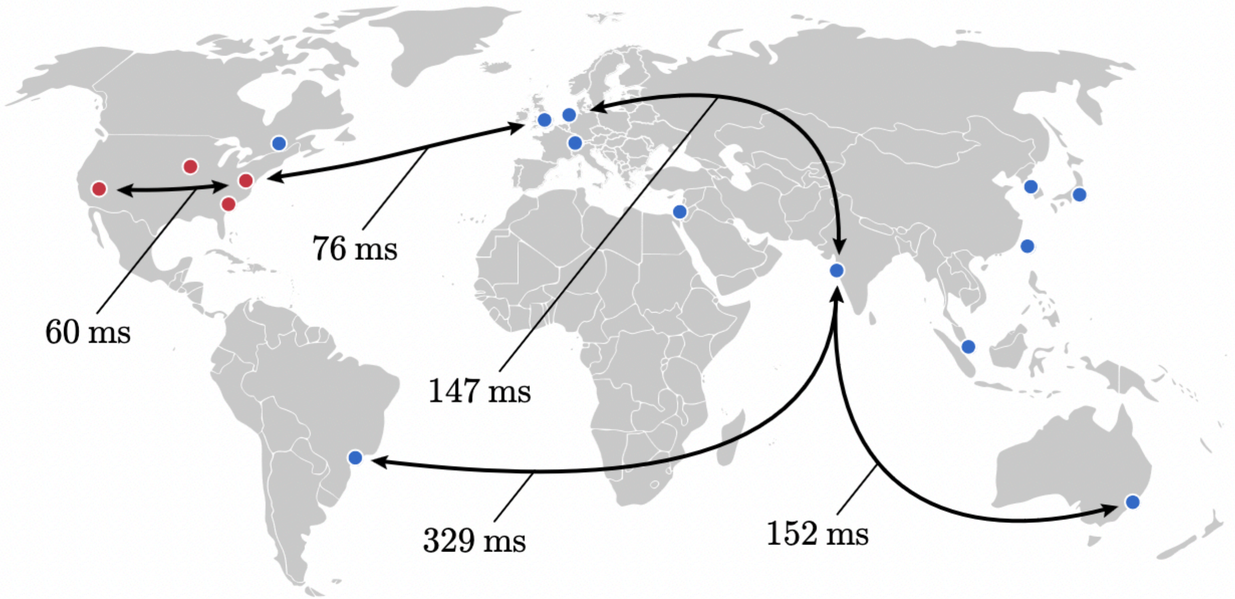
How do we characterize unknown adversaries? How do we formalize intuitive security notions?
What kinds computation can we perform securely in each setting?

4. Other Doors that Computational Security has Opened

The Elephantine Cost in the Room

So far, we have introduce the GMW and BGW protocols. Both are perfectly secure in a hybrid model and both have a round count proportionate to the multiplicative depth of the circuits they compute. We don't know how to do better than that with this flavor of secret-sharing-based protocol.

Latency between Google datacenters, measured by your instructor in 2023.



This is the best case scenario! Hardwired, high-performance internet connections owned by one of the world's richest companies.

Imagine if you were using a very crappy cellphone instead, and you lived on a small pacific island. Or imagine that your only way to connect was via satellite and the nearest downlink was in another country.

Toward Circuit Garbling

We have been hiding all of the intermediate wire values in our computation information theoretically. Instead, what if we give a party *complete information* about everyone's secret inputs, but make sure that it's *computationally infeasible* to learn anything other than the chosen circuit's output. Might this enable us to compute the whole circuit without interacting (after the inputs are exchanged)?

When we realized OT using the DDH assumption, we effectively did this for a single gate. Now we want to do it for an entire circuit.

The protocol we will describe next is even older than GMW. It was originally described by Andrew Yao in a talk in 1986. He didn't write it down, but GMW did one year later.

In the 1980s it was hard
not to get a Turing Award
if you studied cryptography

Toward Circuit Garbling

Before we describe the protocol, recall that in Lecture 4, we proved that perfectly secrecy is impossible if the key is shorter than the message.

Now we have introduced the notion of computational indistinguishability, which allows us to describe distributions that are "just as good as" one another so long as the any algorithm that might distinguish them is PPT.

Intuitively, we can circumvent Shannon's key-length bound if we have some distribution of keys that is computationally indistinguishable from uniform, but nevertheless has short representations.

Pseudorandomness

Intuitively, we can circumvent Shannon's key-length bound if we have some distribution of keys that is computationally indistinguishable from uniform, but nevertheless has short representations.

Definition 1 (Pseudorandom Generator). Let ℓ be a polynomial and let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time function such that for any $\kappa \in \mathbb{N}$ and any $s \in \{0,1\}^\kappa$, $G(s) \in \{0,1\}^{\ell(\kappa)}$.

G is a *pseudorandom generator* if and only if $\ell(\kappa) > \kappa$ and $\forall$ PPT $\mathscr{D}$ $\exists$ negligible ε s.t.

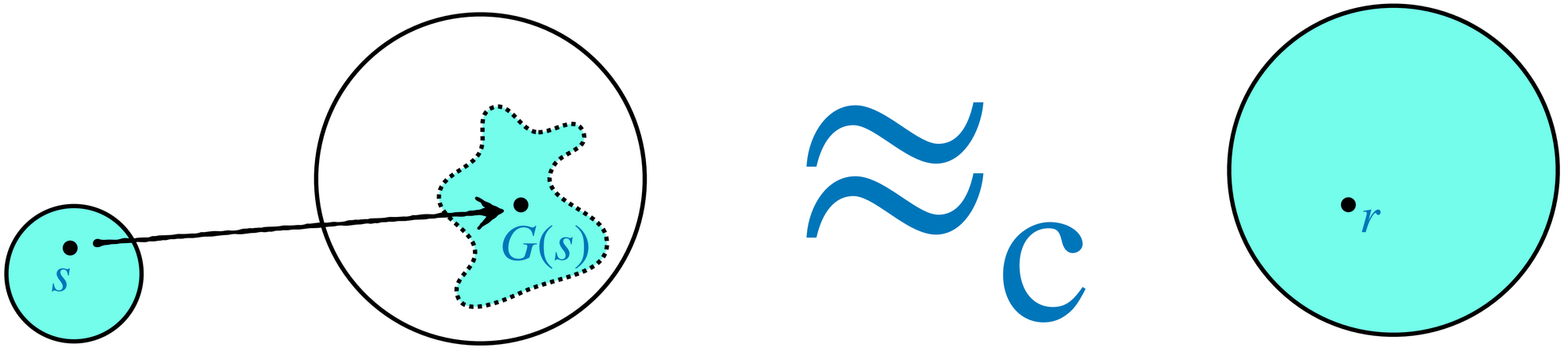
$$\left|\Pr[\mathscr{D}(1^\kappa, G(s)) = 1 : s \leftarrow \{0,1\}^\kappa] - \Pr[\mathscr{D}(1^\kappa, r) = 1 : r \leftarrow \{0,1\}^{\ell(\kappa)}]\right| \leq \varepsilon(\kappa)$$

Pseudorandomness

Definition 1 (Pseudorandom Generator). Let ℓ be a polynomial and let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time function such that for any $\kappa \in \mathbb{N}$ and any $s \in \{0,1\}^\kappa$, $G(s) \in \{0,1\}^{\ell(\kappa)}$.

G is a *pseudorandom generator* if and only if $\ell(\kappa) > \kappa$ and $\forall$ PPT $\mathscr{D}$ $\exists$ negligible ε s.t.

$$\left|\Pr[\mathscr{D}(1^\kappa, G(s)) = 1 : s \leftarrow \{0,1\}^\kappa] - \Pr[\mathscr{D}(1^\kappa, r) = 1 : r \leftarrow \{0,1\}^{\ell(\kappa)}]\right| \leq \varepsilon(\kappa)$$



Uniform over $\{0,1\}^\kappa$ Image of G in $\{0,1\}^{\ell(\kappa)}$ Uniform distribution over $\{0,1\}^{\ell(\kappa)}$

Pseudorandomness

Definition 1 (Pseudorandom Generator). Let ℓ be a polynomial and let $G : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time function such that for any $\kappa \in \mathbb{N}$ and any $s \in \{0,1\}^\kappa$, $G(s) \in \{0,1\}^{\ell(\kappa)}$.

G is a *pseudorandom generator* if and only if $\ell(\kappa) > \kappa$ and $\forall$ PPT $\mathscr{D}$ $\exists$ negligible ε s.t.

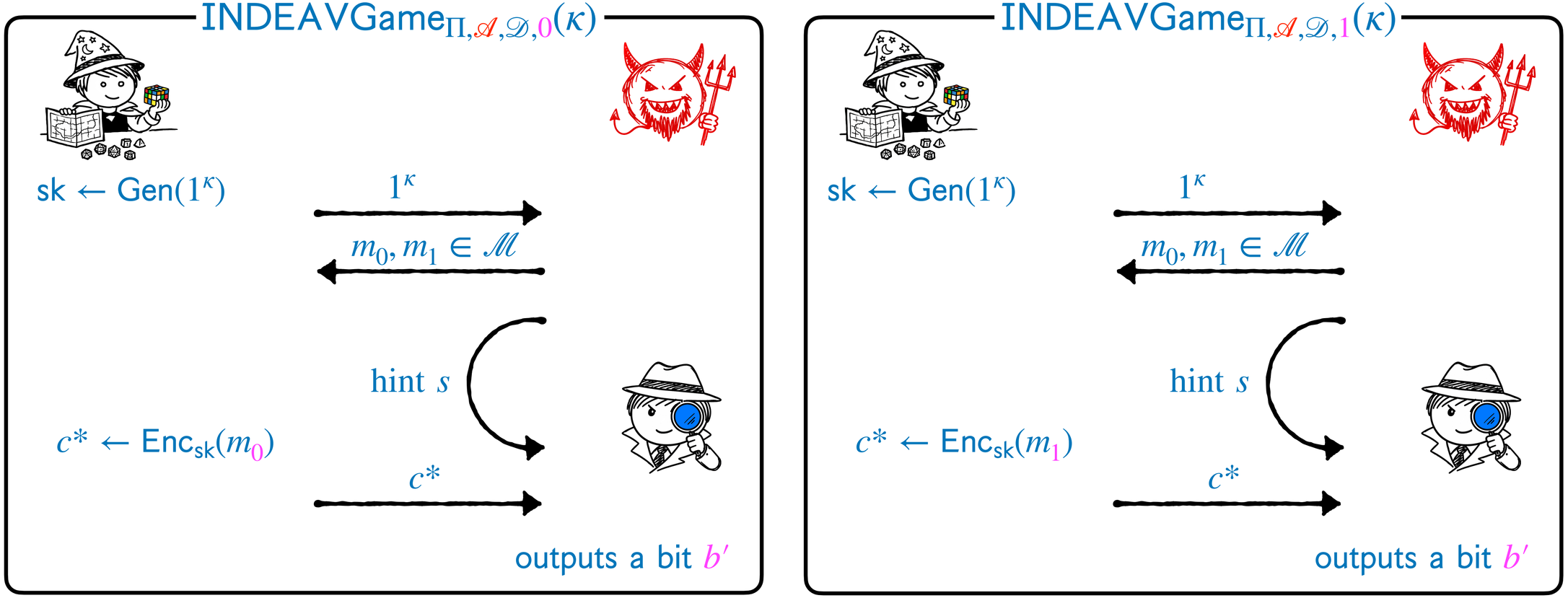
$$\left|\Pr[\mathscr{D}(1^\kappa, G(s)) = 1 : s \leftarrow \{0,1\}^\kappa] - \Pr[\mathscr{D}(1^\kappa, r) = 1 : r \leftarrow \{0,1\}^{\ell(\kappa)}]\right| \leq \varepsilon(\kappa)$$

Imagine if we directly replaced the keys in one-time pad with keys that we sampled using a PRG. We still wouldn't be able to *reuse* keys, so we won't achieve CPA security (defined in Lecture 14).

We also won't achieve perfect secrecy. We need a computational notion of security for one-time encryption.

Definition 2: The IND-EAV Game

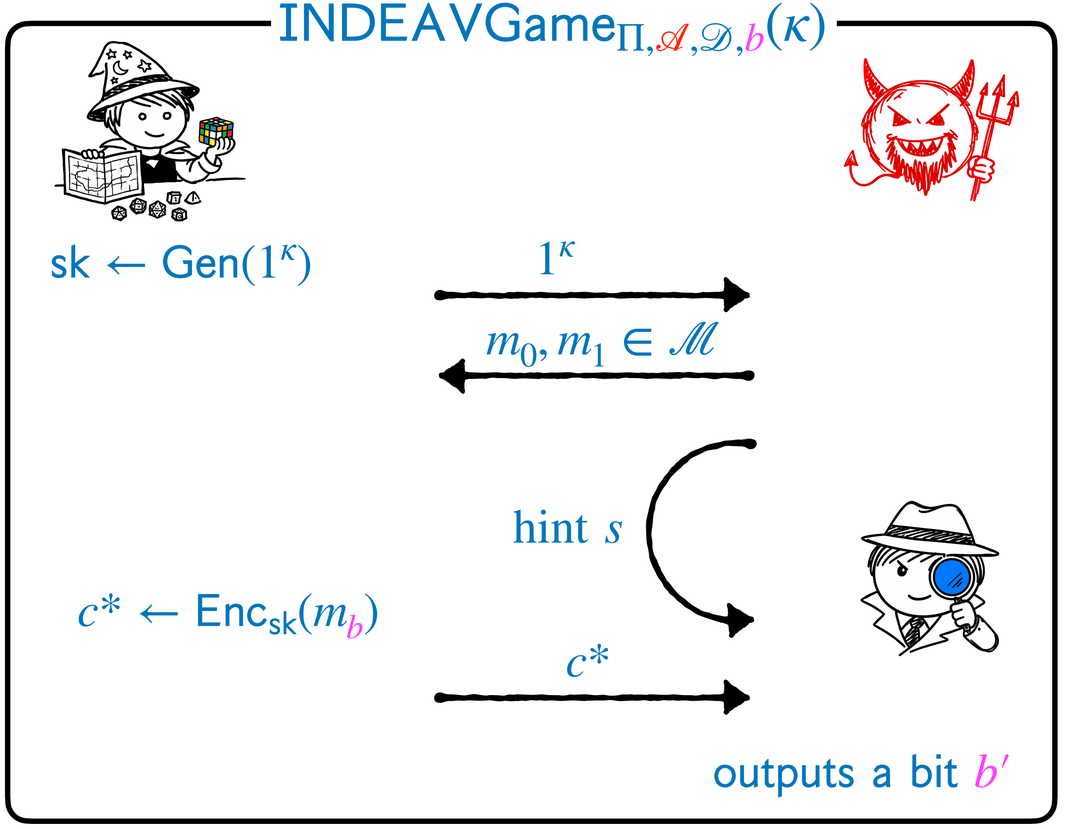
- Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a *symmetric*-key encryption scheme for message space $\mathscr{M}$.
- The *Ciphertext Indistinguishability Eavesdropping Game* $\mathsf{INDEAVGame}_{\Pi,\mathscr{A},\mathscr{D},b}(\kappa)$ for $b \in \{0,1\}$ is played between a challenger and an adversary/distinguisher team. The game's output is the output of the distinguisher.



IND-EAV Security Definition

Definition 3 (IND-EAV Security for SKE):
We say that SKE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has *indistinguishable ciphertexts under Eavesdropping* if $\forall$ PPT $(\mathcal{A}, \mathcal{D})$ $\exists$ a negligible function ε such that $\forall \kappa \in \mathbb{N}$,

$$\left| \Pr\left[\mathsf{INDEAVGame}_{\Pi,\mathcal{A},\mathcal{D},0}(\kappa) = 1\right] - \Pr\left[\mathsf{INDEAVGame}_{\Pi,\mathcal{A},\mathcal{D},1}(\kappa) = 1\right] \right| \leq \varepsilon(\kappa)$$



Note: the only things that distinguish this game from the IND-CPA game defined in Lecture 14 are the fact that Π is symmetric and $\mathcal{A}$ and $\mathcal{D}$ are not allowed to see any encryptions under sk before distinguishing!

Comparing Encryption Security Notions

	Perfect Secrecy	IND-EAV Security	IND-CPA (SKE)	IND-CPA (PKE)
Type	Unbounded Distinguisher	PPT Distinguisher	PPT Distinguisher	PPT Distinguisher
Can $\mathcal{A}$/$\mathcal{D}$ see other ciphertexts?	No	No	Yes, $\mathcal{A}$/$\mathcal{D}$ can query the challenger on any message to get an encryption	Yes, $\mathcal{A}$/$\mathcal{D}$ get pk and can generate encryptions on their own
Key Length Key Reuse?	As long as message No reuse	Security parameter No reuse	Security parameter Reuse is OK	Security parameter Reuse is OK

- In Homework 3, you will hypothesize additional kinds of security that have stronger properties than these…

Constructing IND-EAV Encryption

Definition 4 (Computational OTP). Let ℓ be a polynomial and let $G : \{0,1\}^* \to \{0,1\}^*$ be PRG with output length $\ell(\kappa)$ on any κ-bit input. The computational OTP encryption scheme for $\mathcal{M} = \{0,1\}^{\ell(\kappa)}$ is:

- $\mathsf{Gen}(1^\kappa)$ outputs $k \leftarrow \{0,1\}^\kappa$.
- $\mathsf{Enc}_{\mathsf{sk}}(m)$ outputs $c := m \oplus G(k)$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$ outputs $m' := c \oplus G(k)$.

Theorem 6: Computational OTP is IND-EAV secure.

Proof Idea: Consider a hybrid game that uses OTP instead of Computational OTP. By reduction to the pseudorandomness of G, version b of the hybrid game and version b of the IND-EAV game are computationally indistinguishable. The perfect indistinguishability of the two versions of the hybrid game then yields the theorem.

Constructing PRGs

We can prove that any One-Way Function implies a PRG, but this proof is highly nontrivial and most people never learn it!

In grad crypto, we do a slightly easier proof that special *Injective* One-way Functions imply PRGs, and we prove that you can construct an injective one way function if the Discrete Logarithm Assumption is true. Essentially all cryptographic assumptions imply one-way functions.

To get some intuition, recall that if the DDH problem is hard relative to $\mathcal{G}$, then
$\left\{(g^a, g^b, g^{a\cdot b}) : (\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa), (a,b) \leftarrow \mathbb{Z}_q^2\right\} \approx_c \left\{(g^a, g^b, g^c) : (\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa), (a,b,c) \leftarrow \mathbb{Z}_q^3\right\}$

Notice that if we define $G : \mathbb{Z}_q^2 \to \mathbb{G}^3$ such that $G(a,b) = (g^a, g^b, g^{a\cdot b})$, then we can use an input that is uniform among q^2 possibilities to produce an output that is indistinguishable from uniform among q^3 possibilities.

This isn't *quite* what we want, because we want bits, not group elements, and we haven't dealt with the randomness required to sample the group itself. Nevertheless, it's an intuition.

Constructing PRGs (in Practice)

To get some intuition, recall that if the DDH problem is hard relative to $\mathcal{G}$, then
$\left\{(g^a, g^b, g^{a\cdot b}) : (\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa), (a, b) \leftarrow \mathbb{Z}_q^2\right\} \approx_c \left\{(g^a, g^b, g^c) : (\mathbb{G}, g, q) \leftarrow \mathcal{G}(1^\kappa), (a, b, c) \leftarrow \mathbb{Z}_q^3\right\}$

Notice that if we define $G : \mathbb{Z}_q^2 \to \mathbb{G}^3$ such that $G(a, b) = (g^a, g^b, g^{a\cdot b})$, then we can use an input that is uniform among q^2 possibilities to produce an output that is indistinguishable from uniform among q^3 possibilities.

This isn't *quite* what we want, because we want bits, not group elements, and we haven't dealt with the randomness required to sample the group itself. Nevertheless, it's an intuition.

In practice, we usually use *heuristic* PRGs. Usually these are derived from fixed hash or block-cipher circuits (e.g. SHA or AES) and do not have a proof of security under *any* well-known assumption (indeed, they do not even fulfill the syntactical requirement that they take keys of any length). However, they are often *orders of magnitude* faster than theoretically-sound PRGs.

More generally, "symmetric primitives," which include OWFs, PRGs, and SKE are usually *much* faster in practice than "public-key primitives," including PKE, key exchange, and OT.

Preview of Garbled Circuits

In the next lecture we will describe Yao's Garbled Circuits in detail. For now, let's recall that a boolean gate can be represented by a truth table. So for example, here is the AND gate:

AND ($\wedge$)

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Suppose that we were to pick a *random* string of length κ to encode each possible value on each of the wires.

AND ($\wedge$)

x	y	$x \wedge y$
k_0^x	k_0^y	$k_0^{x \wedge y}$
k_0^x	k_1^y	$k_0^{x \wedge y}$
k_1^x	k_0^y	$k_0^{x \wedge y}$
k_1^x	k_1^y	$k_1^{x \wedge y}$

Now we use the input wire encodings to encrypt the output wire encodings

"Garbled"

$x \wedge y$
$\mathsf{Enc}_{k_0^x}\left(\mathsf{Enc}_{k_0^y}\left(k_0^{x \wedge y}\right)\right)$
$\mathsf{Enc}_{k_0^x}\left(\mathsf{Enc}_{k_1^y}\left(k_0^{x \wedge y}\right)\right)$
$\mathsf{Enc}_{k_1^x}\left(\mathsf{Enc}_{k_0^y}\left(k_0^{x \wedge y}\right)\right)$
$\mathsf{Enc}_{k_1^x}\left(\mathsf{Enc}_{k_1^y}\left(k_1^{x \wedge y}\right)\right)$

If Alice does this and sends the garbled table to Bob, what can he learn?

What if she sends him one of (k_0^x, k_1^x), and he uses OT to choose one of (k_0^y, k_1^y) for himself?

Can you connect this gate to another one?

CS4501 Cryptographic Protocols
Lecture 15: OT from OT, GMW, Pseudorandomness, Garbling

https://jackdoerner.net/teaching/#2026/Spring/CS4501