2-ROUND SECURE MPC FROM INDISTINGUISHABILITY OBFUSCATION

Sanjam Garg, Craig Gentry, Shai Halevi (IBM),
Mariana Raykova (SRI)

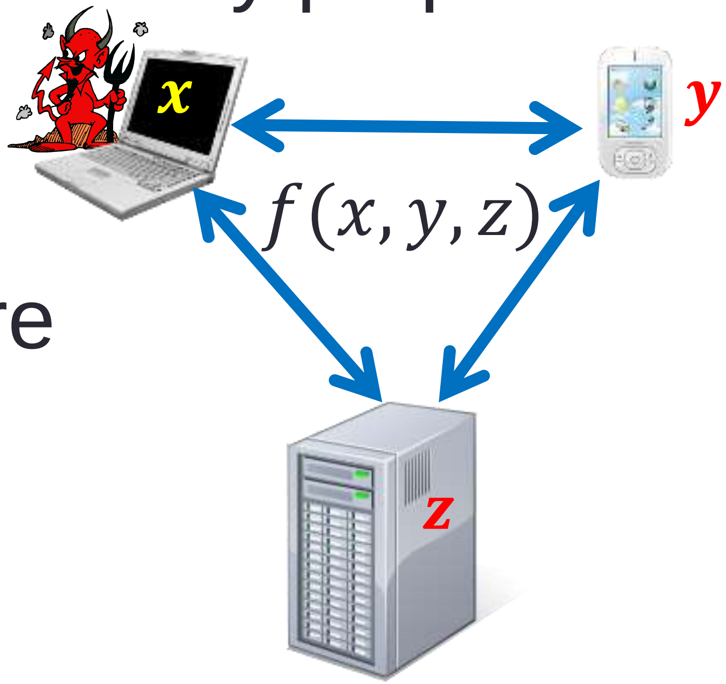
BACKGROUND: SECURE MULTI-PARTY COMPUTATION

Many slides borrowed from Yehuda Lindell

Secure Multiparty Computation

- A set of n parties with private inputs
- Wish to compute on their joint inputs
- While ensuring some security properties
  - Privacy, Correctness,…

- Even if some parties are adversarial



Adversarial behavior

Semi-honest: follows the protocol

- Trying to learn more than what's allowed by inspecting transcript

Malicious: deviates arbitrarily from protocol

- Trying to compromise privacy, correctness, or both

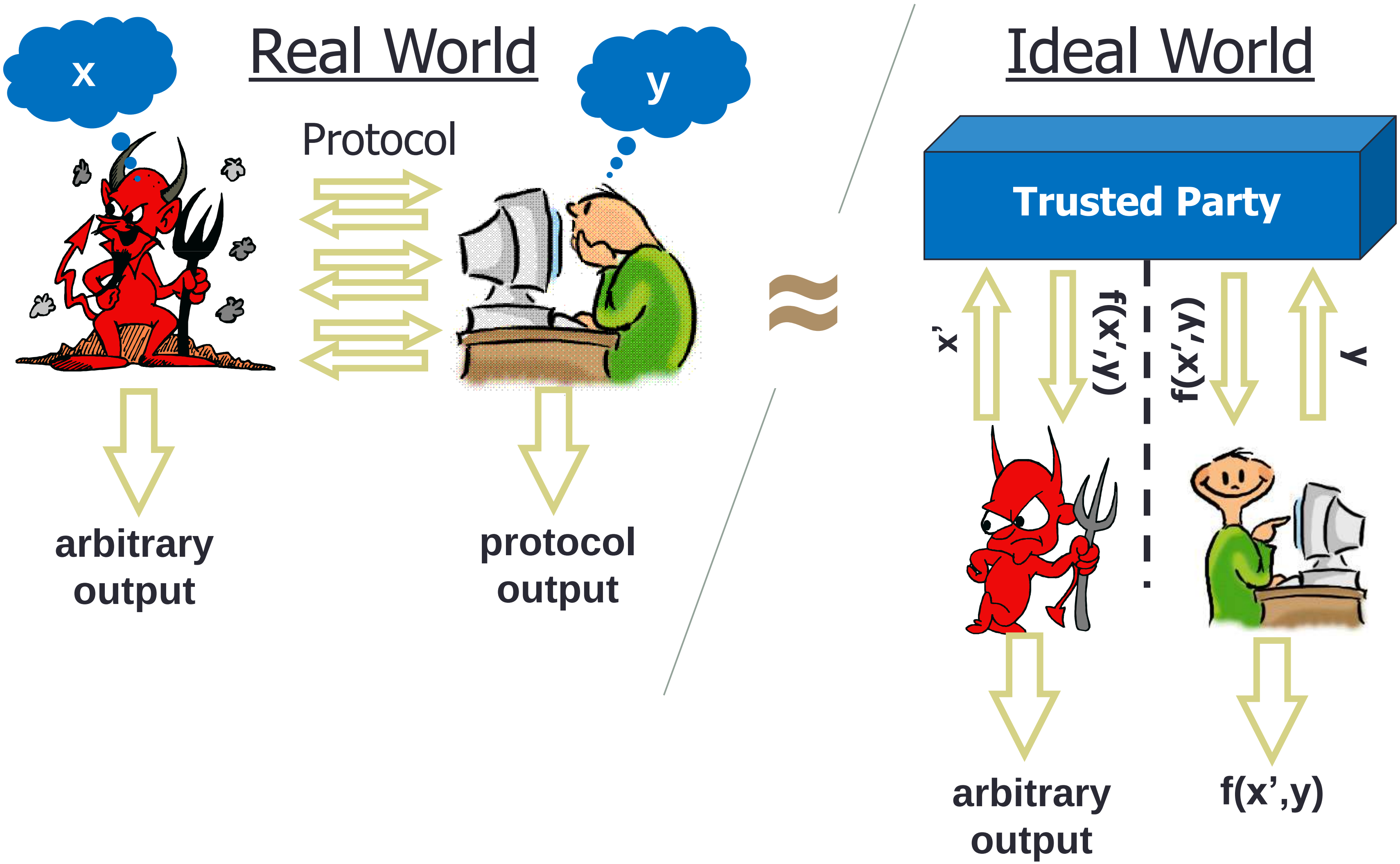
Defining Security: the Ideal/Real Paradigm

- What is the best we could hope for?
  - An incorruptible trusted party
  - All parties send inputs to trusted party
    - over perfectly secure communication lines
  - Trusted party computes output, sends to parties
- This is an ideal world
- What can an adversary do?
  - Just choose its input(s)…

Defining Security: the Ideal/Real Paradigm

- A real-world protocol is secure if it emulates an ideal-world execution
  - Any damage that can happen in the real world can also happen in the ideal world
- Ideal-world adversary cannot do much, so the same is true of the real-world adversary
  - Privacy, correctness, independence of inputs (and more), all hold in the real world

The Ideal/Real Paradigm



The Ideal/Real Paradigm

A n-party protocol π securely realizes the n-input function $f(x_1, \ldots, x_n)$ if

- For every real-world adversary $\boldsymbol{A}$
  - Controlling some bad players, interacting with protocol
- There exists an ideal-world simulator $\boldsymbol{S}$
  - Same bad players, interacting with the trusted party
- s.t. for any environment $\boldsymbol{Z}$ (supplying the inputs):

$$\mathbf{View}_{Z,A}^{real} \approx \mathbf{View}_{Z,S}^{ideal}$$

[GMW86,…] Any f has a secure protocol π_f

- Extensions to "interactive functions" […,C01,…]

Some Specifics of Our "Real World"

- We assume trusted setup (CRS)
  - A random common reference string is chosen honestly, made available to all the players
    - E.g., hard-wired into the protocol implementation
- A broadcast channel is available
  - If I received *msg*, everyone received same *msg*
- The set of bad players is determined before the protocol execution
  - Aka "static corruption model"

Round Complexity of Secure MPC

- Without privacy, one round is enough
  - Everyone broadcast their inputs
- With privacy, need at least two
  - Else, bad guys get access to residual function $f_{fixed\ good\ guys\ inputs}(\vec{x}) = f(fixed\ good\ guys\ inputs, \vec{x})$
  - Can evaluate residual function on many inputs
  - Yields more info on the good guys inputs than what they can get in the ideal world

Round Complexity of Secure MPC

- Can we get 2-round secure computation?
  - Two broadcast rounds after seeing the CRS
- Before this work, best result was 3 rounds
  - [Asharov, Jain, Lopez-Alt, Tromer, Vaikuntanathan, Wichs, Eurocrypt 2012], using threshold (multi-key) FHE
- This work: doing it in two rounds
  - Using heavy tools (*iO*, NIZK)

The Tools We Use

- We start from an Interactive Semi-Honest-Secure Protocol for f
- Compile it into a 2-round protocols using:
  - Indistinguishability Obfuscation
  - Noninteractive Zero-Knowledge (w/ stat. soundness)
  - Chosen-Ciphertext Secure Encryption

Main Tool: Obfuscation

- Make programs “unintelligible” while maintaining their functionality
  - Example from Wikipedia:

```
@P=split//,".URRUU\c8R";@d=split//,"\nrekcah xinU /
lreP rehtona tsuJ";sub p{
@p{"r$p","u$p"}=(P,P);pipe"r$p","u$p";++$p;($q*=2)+
=$f=!fork;map{$P=$P[$f^ord ($p{$_})&6];$p{$_}=/
^$P/ix?$P:close$_}keys%p}p;p;p;p;p;map{$p{$_}=~/^[P
.]/&& close$_}%p;wait
until$?;map{/^r/&&<$_>}%p;$_=$d[$q];sleep
rand(2)if/\S/;print
```

  - Rigorous treatment [Hada’00, BGIRSVY’01,…]
  - Constructions [GGHRSW13,…]

What's "Unintelligible"?

- What we want: can't do much more with obfuscated code than running it on inputs
  - At least: If function depends on secrets that are not apparent in its I/O, then obfuscated code does not reveal these secrets
- [B+01] show that this is impossible:
  - Thm: If PRFs exist, then there exists PRF families $F = \{f_s\}$, for which it is possible to recover s from any circuit that computes f_s.
    - These PRFs are unobfuscatable

What's "Unintelligible"?

- Okay, some function are bad, but not all…
  - Can we get OBF() that does "as well as possible" on every function?
- [B+01] suggested the weaker notion of "indistinguishability obfuscation" (*iO*)
  - Gives the "best-possible" guarantee [GR07]
  - Turns out to suffice for many applications, including ours

Defining Obfuscation

- An efficient public procedure OBF(*)
- Takes as input a program C
  - E.g., encoded as a circuit
- Produce as output another program C'
  - C' computes the same function as C
  - C' at most polynomially larger than C
- Indistinguishability-Obfuscation (*iO*)
  - If C_1, C_2 compute the same function (and $|C_1| = |C_2|$), then $OBF(C_1) \approx OBF(C_2)$

Another Tool: Noninteractive ZK

(slide due to Jens Groth)

Common reference string:
0100…11010

$(x, w) \in R_L$

Statement: $x \in L$

Proof: π $\longrightarrow$

Zero-knowledge:
Nothing but truth revealed

Soundness:
Statement is true

Non Interactive Zero Knowledge

- Proving statement of the form $x \in L$
  - L is an NP language, x is public

NIZK has three algorithms (+ a simulator)

- **CRS generation**: $\sigma \leftarrow \boldsymbol{K}\left(1^k\right)$
- **Proof**: $\pi \leftarrow \boldsymbol{P}(\sigma, x, w)$
- **Verification**: $\boldsymbol{V}(\sigma, x, \pi) = 0/1$
- Simulator: $(\sigma, \tau) \leftarrow \boldsymbol{S}_1\left(1^k\right)$, $\pi \leftarrow \boldsymbol{S}_{\mathbf{2}}(\sigma, \tau, x)$

Non Interactive Zero Knowledge

Perfect completeness: for all $(x, w) \in R_L$

$$\Pr\left[\begin{array}{c}\sigma \leftarrow K\left(1^k\right), \pi \leftarrow P(\sigma, x, w) \\ V(\sigma, x, \pi) = 1\end{array}\right] = 1$$

Statistical soundness:

$$\Pr\left[\begin{array}{c}\sigma \leftarrow K\left(1^k\right) \\ \exists (x, \pi), x \notin L, V(\sigma, x, \pi) = 1\end{array}\right] = negl(k)$$

Computational ZK: for all $(x, w) \in R_L$

$$\left[\sigma \leftarrow K\left(1^k\right), \pi \leftarrow P(\sigma, x, w)\right] \approx^c \left[S(1^k, x)\right]$$

Last Tool: CCA-Secure Encryption

Public-key encryption $(KeyGen, Enc, Dec)$

$$(sk, pk) \leftarrow KeyGen(1^k)$$

<u>Challenger(pk, sk)</u> <u>Adversary(pk)</u>

$$\xleftarrow{\quad c_i \quad}$$

$$\xrightarrow{\quad m_i = Dec_{sk}(c_i) \quad}$$

$$\xleftarrow{\quad m_0^*, m_1^* \quad}$$

$b \leftarrow \{0,1\}$ $\xrightarrow{\quad c^* \leftarrow Enc_{pk}(m_b^*) \quad}$

$$b' \longrightarrow$$

- Adversary wins if c^* not queries and $b' = b$
- Scheme is secure if $\forall A, \Pr[\mathrm{A}\ wins] \lesssim {}^1/_2$

OUR PROTOCOL

Starting Point: Use Obfuscation

- Start from any t-round secure MPC Π
- Consider the next-message functions
  $NextMsg_{x_i,r_i}(\Pi\ transcript\ so\ far) = next\ \Pi\ message\ of\ player\ i$
  - With input, Π-randomness hard-wired in

Starting Point: Use Obfuscation

- Players obfuscate, broadcast, their next-message functions
  - With input, Π-randomness hard-wired in
  - Each player obfuscates one function per round
- Then everyone can locally evaluate the obfuscated functions to get the final output
- But this is a one-round protocol, so it must leak the residual function

Add a Commitment Round

- 1st round: commit to input, Π-randomness
  - Using CCA-secure encryption
- 2nd round: obfuscate next-message functions
  - With input, Π-randomness hard-wired in
  - Also the 1st-round commitments hard-wired in
- We want next-msg-functions to work only if transcript is consistent with commitments
  - This will prevent bad guys from using it with inputs other than ones committed in 1st round

Proofs of Consistency

- $NextMsg'_{x_i, r_i, \mathbf{comms}, \boldsymbol{\sigma}, \boldsymbol{r'_i}}(trans\ \text{so far}, \text{proofs}) =$

$$\left\{\begin{array}{l} \text{verify proofs that } trans \text{ consistent with } comms, \sigma \\ \qquad\qquad \text{If any proof fails output } \perp \\ \qquad\qquad \text{else output (next } \Pi \text{ msg, new proof)} \end{array}\right.$$

  - New-proof generated with randomness r'_i
  - Proves that next-msg was generated by Π
    - on $(trans, x_i, r_i)$, for some x_i, r_i consistent with $comms, \sigma$

- Each party obfuscates, broadcasts $NextMsg'_{x_i, r_i, comms, \sigma, r'_i}$

Is It Secure?

- It would be if we had “ideal obfuscation”
  - “Easy to show” that this is secure when the $NextMsg'$ functions are oracles
  - Essentially since Π+proofs is resettably-secure
  - Key observation: transcript fixed after 1st round
  - This assumes that Π can handle bad randomness
    - Alternatively we can include coin-tossing in the compiler
- But we only have *iO*
  - So we must jump through a few more hoops

Dealing with *iO*

- Change the obfuscated functions as follows:
- $NextMsg''_{x_i, r_i, comms, \sigma, r_i', b, z}(trans \text{ so far, proofs}) =$

$$\begin{cases} \text{verify proofs that } trans \text{ consistent with } comms, \sigma \\ \quad \text{If any proof fails output } \perp \\ \quad \text{else } \begin{cases} \text{if b} = 0 \text{ output (next } \Pi \text{ msg, new proof)} \\ \text{if b} = 1 \text{ output } z \end{cases} \end{cases}$$

- Each player obfuscates t such functions
  - One for every communication round
  - All with same $x_i, r_i, comms, \sigma$, independent r_i''s
  - All with $b = 0, z = 0^\ell$

The Full* Compiler

- CRS: pk of CCA-PKE, σ of NIZK
- 1st round: $P_i(x_i)$ chooses r_i, broadcasts
$$c_i = E_{pk}(i, x_i), d_i = E_{pk}(i, r_i)$$
- 2nd round: P_i chooses $r'_{i,1} \ldots r'_{i,t}$'s, broadcasts
$$F_{i,j} = OBF\left(NextMsg''_{x_i, r_i, \vec{c}, \vec{d}, \sigma, r'_{i,j}, 0, \vec{0}}(\cdot)\right)$$
- Local evaluations: For $j = 1, \ldots, t$, $i = 1, \ldots, n$, use $F_{i,j}(\text{transcript so far, proofs so far})$ to get P_i's j'th message and a proof for it

Complexity, Functionality

- 2 rounds after seeing CRS
- Every $NextMsg''$:
  - Checks at most $t \cdot n$ proofs
  - Computes one protocol message and proves it

  ➔Has complexity at most $poly(k) \cdot Time(\Pi)$
- OBF increases complexity by $poly(k)$ factor

- Correctness follows from correctness of Π and OBF and completeness of proof system

Security

Thm: The compiled protocol UC-securely realizes f against malicious adversaries if

- Π securely realizes f against semi-honest
    - And can tolerate bad randomness
- Proof system is NIZK
- Encryption is CCA secure
- OBF is *iO*

Proof Of Security

- Main idea in the proof:
  - Recall that 1st round fixes the Π-transcript
  - So these two circuits compute the same things:
    - The $NextMsg''$ as constructed in the protocol $(b = 0)$
    - A $NextMsg''$ function with the fixed transcript $(b = 1)$
  - The simulator will use the latter
    - By *iO*, these are indistinguishable.
- Formally: fix adversary A, we describe a simulator, prove its output indistinguishable

The Simulator (1)

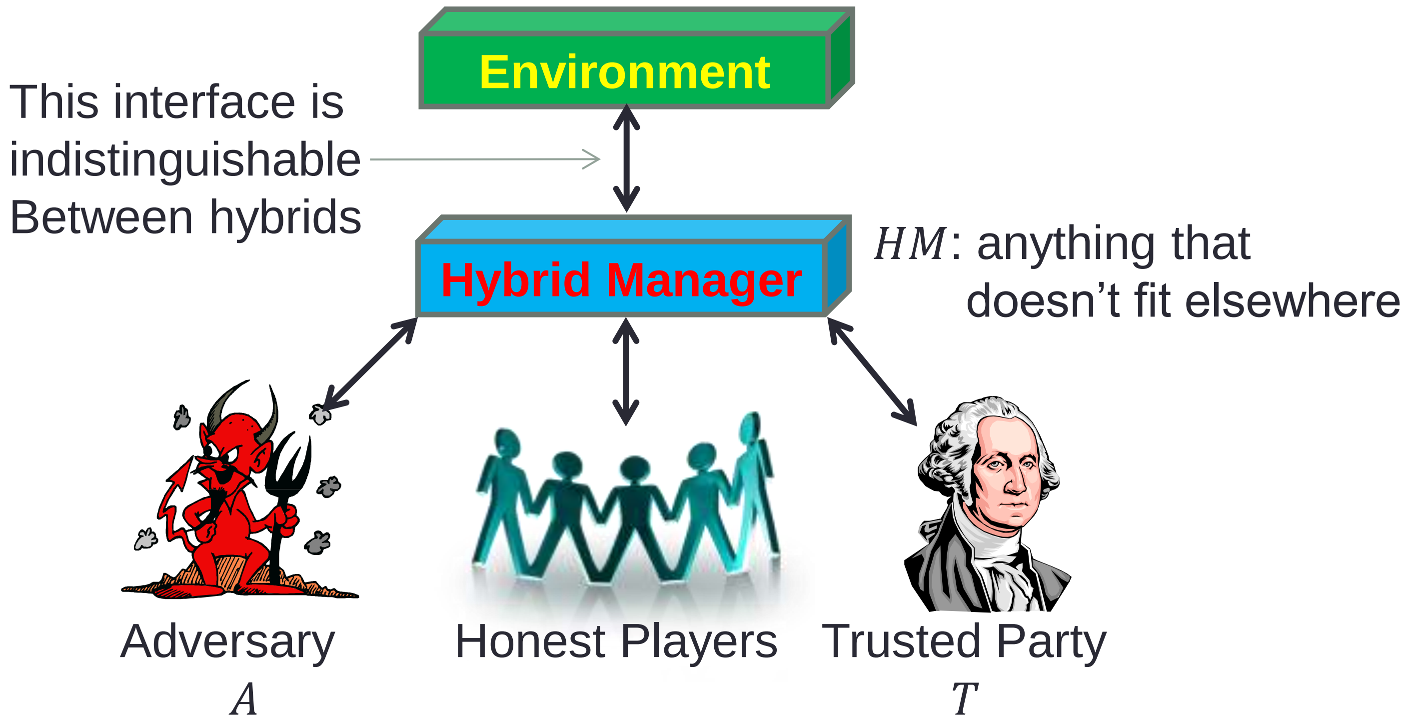
- CRS: $(sk, pk) \leftarrow KeyGen(1^k), (\sigma, \tau) \leftarrow S_1(1^k)$
- Good players' ciphertexts:
$$c_i \leftarrow Enc_{pk}(i, 0), d_i \leftarrow Enc_{pk}(i, 0)$$
- Bad players' ciphertexts:
$$\{c_i, d_i\}_{i \text{ bad}} \leftarrow A\left(pk, \sigma, \{c_i, d_i\}_{i \text{ good}}\right)$$
- Decrypts bad players' c_i, d_i
  - Yields input, randomness for bad players
    - If invalid ciphertext, use default value
- Sends inputs to trusted party, get outputs

The Simulator (2)

- Runs Π-simulator on bad players' (input, output, rand), gets a Π-transcript
- Runs $S_2(\sigma, \tau, \cdot)$ of NIZK, gets proofs for Π-messages of good players
  - Relative to their c_i, d_i's
- Obfuscate $NextMsg''$ for good players
  - Using $x_i = 0, r_i = 0, \text{random } r'_{i,j}\text{'s}$
  - Also using $b = 1, z = (msg, proof)$
    - msg from simulated transcript, $proof$ by NIZK sim.

Real/Ideal Indistinguishability

- We prove indistinguishability by going through several hybrids



Real/Ideal Indistinguishability

- $\boldsymbol{H_1}$ is the real-world game
  - HM runs setup, trusted party is never used
- Lemma: After 1st round, $\exists \leq 1$ Π-transcript for which $\exists$ proofs that would make $NextMsg'''$ output anything other than $\perp$
  - Whp over the CRS, by statistical NIZK soundness
  - Moreover, given sk the HM can efficiently compute that transcript
- Denote that transcript by tr^*

Real/Ideal Indistinguishability

- $\boldsymbol{H_2}$: Obfuscate different functions
  - In H_1 we had $NextMsg'''_{x_i,r_i,\vec{c},\vec{d},\sigma,r'_{i,j},0,\vec{0}}(tr,\mathrm{pfs})$
  - Now we have $NextMsg'''_{\mathbf{0},\mathbf{0},\,\vec{c},\vec{d},\sigma,r'_{i,j},\mathbf{1},\boldsymbol{z}}(tr,\mathrm{pfs})$
    - $z = (msg_z, \mathrm{pf}_z)$ contains the message from tr^*, NIZK proof corresponding to tr^* wrt $\sigma, r'_{i,j}$
- By lemma from above:
  - Both functions output $\bot$ under same conditions
  - If $\text{output} \neq \bot$ then $tr = tr^*$, so both functions output (msg_z, pf_z)

Real/Ideal Indistinguishability

- $\boldsymbol{H_2}$: Obfuscate different functions
  - In H_1 we had $NextMsg'''_{x_i,r_i,\vec{c},\vec{d},\sigma,r'_{i,j},0,\vec{0}}(tr,\mathrm{pfs})$
  - Now we have $NextMsg'''_{\mathbf{0},\mathbf{0},\,\vec{c},\vec{d},\sigma,r'_{i,j},\mathbf{1},\boldsymbol{z}}(tr,\mathrm{pfs})$
    - $z = (msg_z, \mathrm{pf}_z)$ contains the message from tr^*, NIZK proof corresponding to tr^* wrt $\sigma, r'_{i,j}$
- They are functionally identical (whp over CRS)
- By *iO*, their obfuscation is indistinguishable
  - So $H_1 \approx H_2$

Real/Ideal Indistinguishability

- $\boldsymbol{H_3}$: Simulated CRS & NIZKs
  - Indistinguishable by computational ZK
- $\boldsymbol{H_4}$: Encrypt zeroes for honest players instead of inputs & randomness
  - Indistinguishable by security of the PKE
  - Need CCA-security to decrypt A's ciphertexts
    - If adversary copies a good-player ciphertext, then treat it as invalid (since it encrypts the wrong index)

Real/Ideal Indistinguishability

- $\boldsymbol{H_5}$: Use Π-simulator to generate tr^*
  - Send inputs, get outputs from trusted party
  - Indistinguishable by security of Π
  - This is the ideal world, HM is the simulator $\blacksquare$

Reducing Communication Complexity

- The basic construction has communication complexity depends on the complexity of Π
  - Which is at least as large as that of f
- To save communication, use multi-key HE
  - Players encrypt their input, broadcast ctxts
  - Use multi-key HE to evaluate
  - Apply 2^{nd} round of our protocol to the HE decryption function

Questions?