Oblivious Transfer

Recall: garbled circuit \( (f) \): collection of \( 4 \) ciphertexts / gates

garbled input \( (X) \): one wire label per input wire

decoding info \( (d) \): mapping of output wire labels to "true"/"false"

Privacy: \((f, X, d)\) leaks no more than \(f, f(x)\)

\[ \iff \] can be simulated given \((f, f(x))\)

IDEA for 2PC protocol:

\[ (\text{Alice has } x_A, \text{Bob has } x_B) \]

\[ \text{Want to compute } f(x_A, x_B) \]

Alice \((x_A)\)

garble \(f\)

Bob \((x_B)\)

\[ F, d \]

let \(X_A\) be garbled input corresponding to \(x_A\)

\(X_A\)

(Alice doesn't learn \(x_B\))

\(A_0, A_1\) wire labels for \(i\)th input bit

Bob obtains \(X_B\) corresponding to \(x_B\)

Evaluate \(F\) on \(X\)

decode to get \(y = f(x_A, x_B)\)

Oblivious transfer: \((OT)\) // 1-out-of-2 OT

Sender has 2 strings \(S_0, S_1\)

Receiver has "choice bit" \(b \in \{0, 1\}\)

Receiver learns (only) \(S_b\)

Sender learns nothing

Security: receiver's view can be simulated given only \((b, S_b)\)
Claim: this 2PC protocol is secure against semi-honest adversaries (views can be simulated given only input & y)

Alice:
- view contains:
  - messages in OT subprotocols
  - msg y from Bob

Bob:
- view contains:
  - F, X_A, d from Alice
  - msgs from OT protocol

Simulatable given y?
- simulatable given nothing
- because OT leaks nothing to sender
- easy: y is given

Simulatable?
- privacy of gambling
  - ⇒ simulate (F,X,d) given y
  - simulatable given received input & prescribed output
  - ⇒ input = Bob's bits of X_B
  - output = X_B wire labels

What if an adversary is malicious?

Bob could send wrong y in last message
- (causes A to output something inconsistent)

Alice sends A_1, A_2 to OT : effectively fixing Bob's input to 11

Alice sends (A_0, junk) to OT for first of Bob's inputs
- if Bob's input starts w/ 0 and circuit is evaluated fine
- else Bob can't open any of the 4 ciphertexts reports an error maybe?
  - (leaks 1st bit of Bob's input !!)
How to get OT?

Even-Goldreich-Lempel '85

(semi-honest secure)

Uses public-key cryptosystem

**sender** $(s_0, s_1)$

\[ c_0 = \text{Enc}(pk_0, s_0) \]

\[ c_1 = \text{Enc}(pk_1, s_1) \]

**receiver** $(b)$:

Choose $(pk, sk) \leftarrow \text{KeyGen}$

[Sample $pk'$ to look like a public key, w/o knowing secret key]

\[ pk_b = pk \]

\[ pk_{1-b} = pk' \]

\[ s_b = \text{Dec}(sk, c_b) \]