Multiparty Computation

Principle

- There is a (randomized) function $f : (\{0,1\}^\ell)^n \longrightarrow (\{0,1\}^\ell)^n$.
- There are n parties, $P_1, \ldots, P_n$.
  - Some of them may be adversarial.
  - Two forms of adversarial behaviour:
    - Semi-honest — will follow the protocol as prescribed, but tries to deduce extra information from what it sees.
    - Malicious — will not necessarily follow the protocol.
- Party P_i has the bit-string $x_i \in \{0,1\}^\ell$.
- Party P_i wants to learn y_i, where

$$(y_1, \ldots, y_n) = f(x_1, \ldots, x_n) \ .$$

- No party P_i may learn anything beyond x_i and y_i.

Examples

- Two millionaires want to determine who is richer.
  - Revealing one's net worth to the other party means losing one's face if the other party turns out to be much richer.
- Alice and Bob are considering to start dating each other.
  - Revealing that one is interesting in dating means losing one's face if the other party is not interested.
- Three cryptographers are dining in a restaurant. The waiter informs them that the bill has already been payed. The cryptographers want to know whether the payer was one of them (who wants to remain anonymous) or the NSA.
- Everything else...

Pieces

- The number of parties n.
  - $n = 2$
  - $n \geq 3$
- A function f, represented as a Boolean circuit.
  - Deterministic or randomized.
  - Common output or separate outputs.
- A n-party protocol Π. Two main techniques:
  - "Garbled circuits"
  - Secret-sharing the values on wires.
- The adversary (a coalition of parties)
  - maximum size
  - semi-honest or malicious
  - non-adaptive or adaptive

Security definition

- Consider the deterministic two-party, semi-honest case.
- A protocol Π securely evaluates the function $f(x_1, x_2)$ if there exist PPT simulators S_1 and S_2, such that for all x_1 and x_2:
  - The distribution of $S_1(x_1, f_1(x_1, x_2))$ is indistinguishable from the view of the first party in the execution of $\Pi(x_1, x_2)$.
  - The distribution of $S_2(x_2, f_2(x_1, x_2))$ is indistinguishable from the view of the second party in the execution of $\Pi(x_1, x_2)$.

Security definition

- Consider the randomized two-party, semi-honest case.
- A protocol Π securely evaluates the randomized function $f(x_1, x_2)$ if there exist PPT simulators S_1 and S_2, such that for all x_1 and x_2: The following two distributions are indistinguishable (for $i \in \{1, 2\}$):

  1. First distribution:
     - sample $(y_1, y_2) \leftarrow f(x_1, x_2)$;
     - run $S_i(x_i, y_1, y_2)$.
  2. Second distribution:
     - sample $tr \leftarrow \Pi(x_1, x_2)$;
     - take $(\textsc{View}_1(tr), \textsc{Result}_2(tr))$.

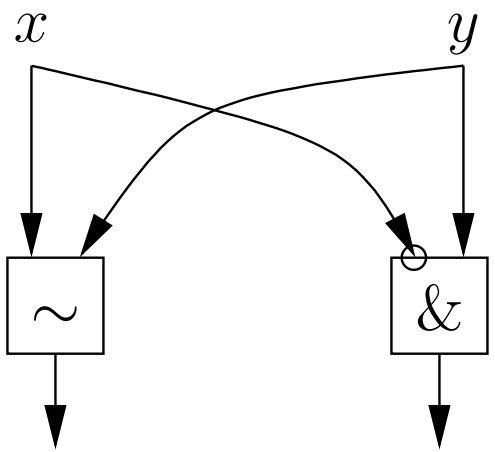
Exercise. Why is the simulator given the output of the other party, too? Compared to deterministic case, what else is there to protect?

Exercises

- Show that the secure evaluation of a randomized f is reducible to the secure evaluation of some deterministic f'.
- Show that the secure evaluation of some f with separate outputs for all parties is reducible to the secure evaluation of some f with common output.

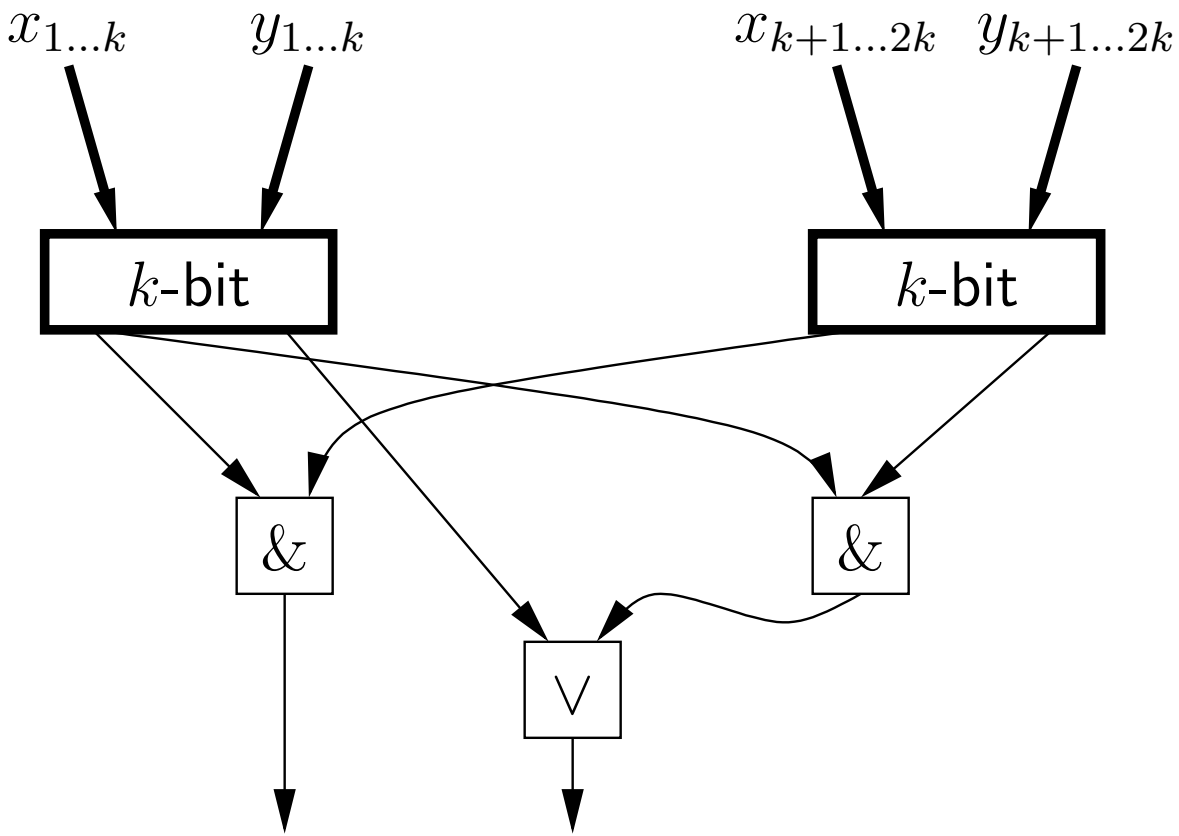
Example functionality

$f(x, y) = (x \stackrel{?}{=} y, x \stackrel{?}{<} y)$ (common output).
For one-bit x and y:



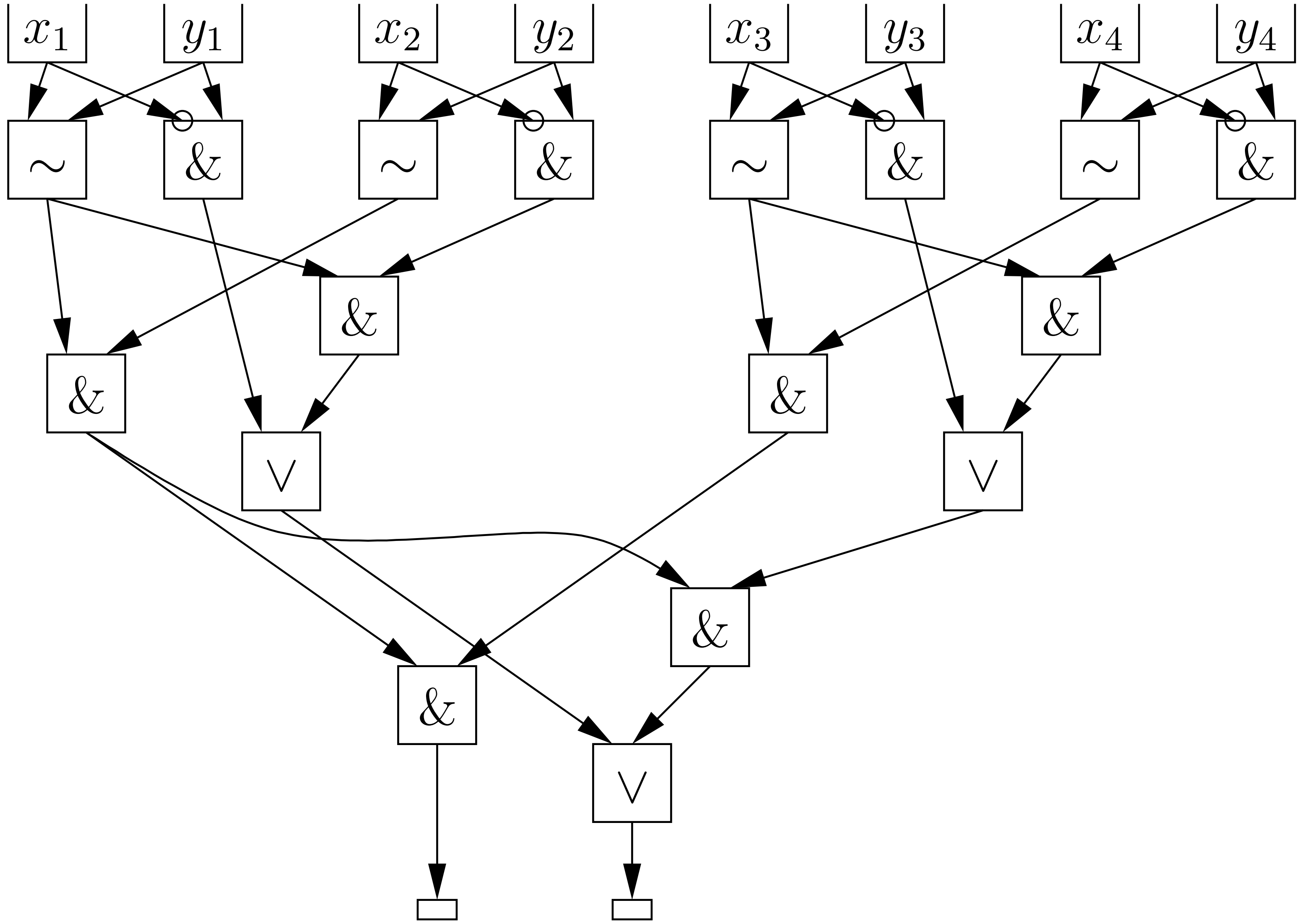
Example functionality

For $2k$-bit x and y:



Example functionality

For 4-bit x and y:



Evaluating a circuit

- Each internal gate g is determined by the function it computes:

0	0	$g(00)$
0	1	$g(01)$
1	0	$g(10)$
1	1	$g(11)$

- The values of input gates are the bits of x and y.
- The value of an internal gate is found from its inputs.
  - Computed from top to bottom.
- The value of an output gate is its input.

Garbling a circuit

- For each input and internal gate g, generate two (symmetric) encryption keys k_g^0 and k_g^1.
- Let g be an internal gate. Let g_1 and g_2 provide the two inputs to g. Compute

$$\left\{\left\{k_g^{g(00)}\right\}_{k_{g_2}^0}^{r_2}\right\}_{k_{g_1}^0}^{r_1}, \left\{\left\{k_g^{g(01)}\right\}_{k_{g_2}^1}^{r_4}\right\}_{k_{g_1}^0}^{r_3}, \left\{\left\{k_g^{g(10)}\right\}_{k_{g_2}^0}^{r_6}\right\}_{k_{g_1}^1}^{r_5}, \left\{\left\{k_g^{g(11)}\right\}_{k_{g_2}^1}^{r_8}\right\}_{k_{g_1}^1}^{r_7}$$

- The encoding of an internal gate g is a random permutation of these four values.
- Let g provide the input to some output gate. The encoding of this output gate is $(\{0\}_{k_g^0}, \{1\}_{k_g^1})$ or $(\{1\}_{k_g^1}, \{0\}_{k_g^0})$.
- The encoding of a circuit maps each internal and output gate to its encoding.

Evaluating a garbled circuit

- Let the input gates be $g_1, \ldots, g_\ell$. Let the input be $b_1 \cdots b_\ell$.
- Somehow obtain $k_{g_1}^{b_1}, \ldots, k_{g_\ell}^{b_\ell}$. Do not obtain $k_{g_1}^{\neg b_1}, \ldots, k_{g_\ell}^{\neg b_\ell}$.
- Let g be an internal gate.
  - Let (m_1, m_2, m_3, m_4) be its encoding.
  - Let g' and g'' provide the inputs to g.
  - Let us know a key k' corresponding to g'.
  - Let us know a key k'' corresponding to g''.
  - Try to decrypt: compute $D_{k''}(D_{k'}(m_i))$ for $1 \leq i \leq 4$.
  - Let k be the result of successful decryption. This key corresponds to g.
- At an output gate we can decrypt one of the ciphertexts, giving us either the bit 0 or the bit 1.

The protocol Π

- Both parties have agreed on the circuit that computes f.
- Party P_1 prepares the garbled circuit and sends it to P_2.
- P_1 sends to P_2 the keys $k_1^{x_1}, \ldots, k_\ell^{x_\ell}$ corresponding to its input x going into the input gates $g_1, \ldots, g_\ell$.
- Party P_1 and P_2 run a protocol, resulting in
  - P_2 learning the keys $k_{\ell+1}^{y_1}, \ldots, k_{2\ell}^{y_\ell}$ corresponding to its input y going into the input gates $g_{\ell+1}, \ldots, g_{2\ell}$;
  - P_1 not learning anything new at all.
- P_2 evaluates the garbled circuit, eventually learning $f(x, y)$.
- P_2 sends $f(x, y)$ back to P_1.

Oblivious transfer

- A special case of two-party computation.
- P_1 (sender) has ℓ-bit strings $m_1, \ldots, m_n$.
- P_2 (receiver) has an integer $i \in \{1, \ldots, n\}$.
- P_2 should learn m_i and nothing else. P_1 should learn nothing.

Oblivious transfer

- A special case of two-party computation.
- P_1 (sender) has ℓ-bit strings $m_1, \ldots, m_n$.
- P_2 (receiver) has an integer $i \in \{1, \ldots, n\}$.
- P_2 should learn m_i and nothing else. P_1 should learn nothing.
- Let E be a trapdoor permutation of some set X including $\{0,1\}^\ell$.
  - for example RSA.
- Let (k_e, k_d) be the public and secret key of P_1.
- P_2 randomly chooses $r_1, \ldots, r_n \in X$. He defines

$$z_j := \begin{cases} r_j, & \text{if } j \neq i \\ E_{k_e}(r_j), & \text{if } j = i \end{cases}$$

  and sends $(z_1, \ldots, z_n)$ to P_1.
- P_1 computes $w_j := m_j \boxplus E_{k_d}^{-1}(z_j)$ and sends $(w_1, \ldots, w_n)$ to P_2.
  - $\boxplus$ — a group operation on X
- P_2 finds m_i as $w_i \boxplus r_i^{-1}$.

Exercise

Show that the preceeding protocol securely performs oblivious transfer in the presence of semi-honest adversaries. I.e. construct the simulators.

Exercise

Show that the preceeding protocol securely performs oblivious transfer in the presence of semi-honest adversaries. I.e. construct the simulators.

- This simulation is not quite correct, though.
- $E_{k_e}(r)$ reveals some information about r.
- But each trapdoor permutation has a hardcore bit B.
- If $m_1, \ldots, m_n$ are just 1 bit long, then the sender may define $w_j := m_j \oplus B(E_{k_d}^{-1}(z_j))$. The receiver recovers m_i as $w_i \oplus B(r_i)$.

Correctness of the protocol Π

- Trace of the first party:
  - Random coins for all the keys and encryptions.
  - Traces of the OT-protocol as the sender.
  - The final result.
- Trace of the second party:
  - A garbled circuit and the keys corresponding to the input bits.
  - Traces of the OT-protocol as the receiver.

Exercise. Construct a simulator for the first party.

Simulator for P_2

- For each input or internal gate g, generate two keys k_g, $\tilde{k}_g$.
- The simulated encoding of the gate $g \leftarrow g_1, g_2$ is

$$\{\{k_g\}^{r_2}_{k_{g_2}}\}^{r_1}_{k_{g_1}}, \{\{k_g\}^{r_4}_{\tilde{k}_{g_2}}\}^{r_3}_{k_{g_1}}, \{\{k_g\}^{r_6}_{k_{g_2}}\}^{r_5}_{\tilde{k}_{g_1}}, \{\{k_g\}^{r_8}_{\tilde{k}_{g_2}}\}^{r_7}_{\tilde{k}_{g_1}}$$

- The simulated encoding of an output gate $\leftarrow g$ is $(\{z\}^{r_1}_{k_g}, \{z\}^{r_2}_{\tilde{k}_g})$, where z is the output bit corresponding to this gate.
- The simulated trace consists of
  - simulated garbled circuit;
  - the keys $k_1, \ldots, k_\ell$;
  - Simulated traces of the OT-protocol as the receiver, resulting in the keys $k_{\ell+1}, \ldots, k_{2\ell}$.

Example — comparing one bit

Let $x_0 = 0$, $y_0 = 1$. Then the output bits are $(0, 1)$. The real view is

$$k_1^0, k_2^1, \{\{\{k_3^1\}_{k_2^0}^{r_2}\}_{k_1^0}^{r_1}, \{\{k_3^0\}_{k_2^1}^{r_4}\}_{k_1^0}^{r_3}, \{\{k_3^0\}_{k_2^0}^{r_6}\}_{k_1^1}^{r_5}, \{\{k_3^1\}_{k_2^1}^{r_8}\}_{k_1^1}^{r_7}\},$$
$$\{\{\{k_4^0\}_{k_2^0}^{r_{10}}\}_{k_1^0}^{r_9}, \{\{k_4^1\}_{k_2^1}^{r_{12}}\}_{k_1^0}^{r_{11}}, \{\{k_4^0\}_{k_2^0}^{r_{14}}\}_{k_1^1}^{r_{13}}, \{\{k_4^0\}_{k_2^1}^{r_{16}}\}_{k_1^1}^{r_{15}}\},$$
$$\{\{0\}_{k_3^0}^{r_{17}}, \{1\}_{k_3^1}^{r_{18}}\}, \{\{0\}_{k_4^0}^{r_{19}}, \{1\}_{k_4^1}^{r_{20}}\}$$

The simulated view is

$$k_1, k_2, \{\{\{k_3\}_{k_2}^{r_2}\}_{k_1}^{r_1}, \{\{k_3\}_{\tilde{k}_2}^{r_4}\}_{k_1}^{r_3}, \{\{k_3\}_{k_2}^{r_6}\}_{\tilde{k}_1}^{r_5}, \{\{k_3\}_{\tilde{k}_2}^{r_8}\}_{\tilde{k}_1}^{r_7}\},$$
$$\{\{\{k_4\}_{k_2}^{r_{10}}\}_{k_1}^{r_9}, \{\{k_4\}_{\tilde{k}_2}^{r_{12}}\}_{k_1}^{r_{11}}, \{\{k_4\}_{k_2}^{r_{14}}\}_{\tilde{k}_1}^{r_{13}}, \{\{k_4\}_{\tilde{k}_2}^{r_{16}}\}_{\tilde{k}_1}^{r_{15}}\},$$
$$\{\{0\}_{k_3}^{r_{17}}, \{0\}_{\tilde{k}_3}^{r_{18}}\}, \{\{1\}_{k_4}^{r_{19}}, \{1\}_{\tilde{k}_4}^{r_{20}}\}$$

The real pattern

The real view is

$$k_1^0, k_2^1, \{\{\{k_3^1\}_{k_2^0}^{r_2}\}_{k_1^0}^{r_1}, \{\{k_3^0\}_{k_2^1}^{r_4}\}_{k_1^0}^{r_3}, \{\{k_3^0\}_{k_2^0}^{r_6}\}_{k_1^1}^{r_5}, \{\{k_3^1\}_{k_2^1}^{r_8}\}_{k_1^1}^{r_7}\},$$
$$\{\{\{k_4^0\}_{k_2^0}^{r_{10}}\}_{k_1^0}^{r_9}, \{\{k_4^1\}_{k_2^1}^{r_{12}}\}_{k_1^0}^{r_{11}}, \{\{k_4^0\}_{k_2^0}^{r_{14}}\}_{k_1^1}^{r_{13}}, \{\{k_4^0\}_{k_2^1}^{r_{16}}\}_{k_1^1}^{r_{15}}\},$$
$$\{\{0\}_{k_3^0}^{r_{17}}, \{1\}_{k_3^1}^{r_{18}}\}, \{\{0\}_{k_4^0}^{r_{19}}, \{1\}_{k_4^1}^{r_{20}}\}$$

Its pattern is

$$k_1^0, k_2^1, \{\{\square^{r_2}\}_{k_1^0}^{r_1}, \{\{k_3^0\}_{k_2^1}^{r_4}\}_{k_1^0}^{r_3}, \square^{r_5}, \square^{r_7}\},$$
$$\{\{\square^{r_{10}}\}_{k_1^0}^{r_9}, \{\{k_4^1\}_{k_2^1}^{r_{12}}\}_{k_1^0}^{r_{11}}, \square^{r_{13}}, \square^{r_{15}}\},$$
$$\{\{0\}_{k_3^0}^{r_{17}}, \square^{r_{18}}\}, \{\square^{r_{19}}, \{1\}_{k_4^1}^{r_{20}}\}$$

The simulated pattern

The simulated view is

$$k_1, k_2, \{\{\{k_3\}^{r_2}_{k_2}\}^{r_1}_{k_1}, \{\{k_3\}^{r_4}_{\tilde{k}_2}\}^{r_3}_{k_1}, \{\{k_3\}^{r_6}_{k_2}\}^{r_5}_{\tilde{k}_1}, \{\{k_3\}^{r_8}_{\tilde{k}_2}\}^{r_7}_{\tilde{k}_1}\},$$
$$\{\{\{k_4\}^{r_{10}}_{k_2}\}^{r_9}_{k_1}, \{\{k_4\}^{r_{12}}_{\tilde{k}_2}\}^{r_{11}}_{k_1}, \{\{k_4\}^{r_{14}}_{k_2}\}^{r_{13}}_{\tilde{k}_1}, \{\{k_4\}^{r_{16}}_{\tilde{k}_2}\}^{r_{15}}_{\tilde{k}_1}\},$$
$$\{\{0\}^{r_{17}}_{k_3}, \{0\}^{r_{18}}_{\tilde{k}_3}\}, \{\{1\}^{r_{19}}_{k_4}, \{1\}^{r_{20}}_{\tilde{k}_4}\}$$

Its pattern is

$$k_1, k_2, \{\{\{k_3\}^{r_2}_{k_2}\}^{r_1}_{k_1}, \{\square^{r_4}\}^{r_3}_{k_1}, \square^{r_5}, \square^{r_7}\},$$
$$\{\{\{k_4\}^{r_{10}}_{k_2}\}^{r_9}_{k_1}, \{\square^{r_{12}}\}^{r_{11}}_{k_1}, \square^{r_{13}}, \square^{r_{15}}\},$$
$$\{\{0\}^{r_{17}}_{k_3}, \square^{r_{18}}\}, \{\{1\}^{r_{19}}_{k_4}, \square^{r_{20}}\}$$

Compare the patterns

The real pattern

$$k_1^0, k_2^1, \{\{\square^{r_2}\}_{k_1^0}^{r_1}, \{\{k_3^0\}_{k_2^1}^{r_4}\}_{k_1^0}^{r_3}, \square^{r_5}, \square^{r_7}\},$$
$$\{\{\square^{r_{10}}\}_{k_1^0}^{r_9}, \{\{k_4^1\}_{k_2^1}^{r_{12}}\}_{k_1^0}^{r_{11}}, \square^{r_{13}}, \square^{r_{15}}\}, \{\{0\}_{k_3^0}^{r_{17}}, \square^{r_{18}}\}, \{\square^{r_{19}}, \{1\}_{k_4^1}^{r_{20}}\}$$

The simulated pattern

$$k_1, k_2, \{\{\{k_3\}_{k_2}^{r_2}\}_{k_1}^{r_1}, \{\square^{r_4}\}_{k_1}^{r_3}, \square^{r_5}, \square^{r_7}\},$$
$$\{\{\{k_4\}_{k_2}^{r_{10}}\}_{k_1}^{r_9}, \{\square^{r_{12}}\}_{k_1}^{r_{11}}, \square^{r_{13}}, \square^{r_{15}}\}, \{\{0\}_{k_3}^{r_{17}}, \square^{r_{18}}\}, \{\{1\}_{k_4}^{r_{19}}, \square^{r_{20}}\}$$

They are equal up to renaming.

In general, the real and simulated garbled circuits will be indistinguishable.

Sharing the contents of wires

- Assume that the only operations of circuits are $\oplus$ and $\&$.
  - These are the addition and multiplication in the field $\mathbb{Z}_2$.
  - They, together with the constant 1, are sufficient to express any functionality.
    - Let there be an extra 1-gate that takes no inputs.
- During the protocol, the parties compute for all gates g the values a_g^1 and a_g^2, such that
  - the real value computed by g is $a_g^1 + a_g^2$;
  - P_1 knows a_g^1 and P_2 knows a_g^2.
- The protocol is well-suited for functionalities with separate outputs.
- At the end of the protocol, P_1 will send to P_2 the values a_g^1 corresponding to the output gates g of P_2.
- P_2 behaves similarly.

The protocol

- **Sharing the inputs**
  - For P_i's input b at the input gate g generate a random bit r and send (g, r) to the other party. Let $a_g^i = b + r$.
  - For the 1-gate g let P_1 generate a random bit r and send (g, r) to P_2. Let $a_g^1 = r + 1$.
  - When P_i receives (g, r) from the other party, set $a_g^i = r$.
- **Evaluating an addition gate** Let $g = g_1 + g_2$. Define $a_g^i = a_{g_1}^i + a_{g_2}^i$.
- **Communicating the outputs** If g is an output gate for P_i, then the other party P_j will send (g, a_g^j) to P_i once he has it. P_i outputs $a_g^i + a_g^j$ as the output of that gate.

Evaluating a multiplication gate

- Let $g = g_1 \cdot g_2 = (a^1_{g_1} + a^2_{g_1}) \cdot (a^1_{g_2} + a^2_{g_2})$.
- We'll define a protocol for finding a^1_g and a^2_g, such that
  - P_i does not learn anything besides a^i_g;
  - a^i_g is uniformly distributed;
  - $a^1_g + a^2_g = a_g$.

Evaluating a multiplication gate

- Let $g = g_1 \cdot g_2 = (a^1_{g_1} + a^2_{g_1}) \cdot (a^1_{g_2} + a^2_{g_2})$.
- We'll define a protocol for finding a^1_g and a^2_g, such that
  - P_i does not learn anything besides a^i_g;
  - a^i_g is uniformly distributed;
  - $a^1_g + a^2_g = a_g$.
- a^1_g is picked uniformly from $\{0, 1\}$.
- P_1 defines

$$m_1 = a^1_g + a^1_{g_1} a^1_{g_2} \qquad m_3 = a^1_g + (a^1_{g_1} + 1) a^1_{g_2}$$
$$m_2 = a^1_g + a^1_{g_1} (a^1_{g_2} + 1) \qquad m_4 = a^1_g + (a^1_{g_1} + 1)(a^1_{g_2} + 1)$$

- P_2 defines $a^2_g = m_{2a^2_{g_1} + a^2_{g_2} + 1}$.
  - Use oblivious transfer to transmit that m to P_2.

Exercise. Correctness and security?

Exercise

How can one party simulate the computation of this protocol?

Multi-party semi-honest case

- A protocol Π securely evaluates the function $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ if
  - there exists a PPT simulator S, such that
  - for each $I = \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$
  - for all $x_1, \ldots, x_n$
  - the distribution $S(I, (x_{i_1}, \ldots, x_{i_m}), (y_{i_1}, \ldots, y_{i_m}))$ equals
  - the joint view of the parties $P_{i_1}, \ldots, P_{i_m}$ in the execution of $\Pi(x_1, \ldots, x_n)$.

The protocol

- Most steps of the two-party case easily generalize to the multi-party case.
- How about multiplication?

The protocol

- Most steps of the two-party case easily generalize to the multi-party case.
- How about multiplication?
- We have $a_{g_1}^1, \ldots, a_{g_1}^n, a_{g_2}^1, \ldots, a_{g_2}^n$ with $\sum_j a_{g_i}^j = a_{g_i}$.
- We want $a_g^1, \ldots, a_g^n$ that sum up to $a_{g_1} \cdot a_{g_2}$.

$$\left(\sum_{j=1}^{n} a_{g_1}^j\right) \cdot \left(\sum_{j=1}^{n} a_{g_2}^j\right) = \sum_{j=1}^{n} a_{g_1}^j a_{g_2}^j + \sum_{1 \le i < j \le n} (a_{g_1}^i a_{g_2}^j + a_{g_2}^i a_{g_1}^j) =$$

$$(1 - (n-1)) \sum_{j=1}^{n} a_{g_1}^j a_{g_2}^j + \sum_{1 \le i < j \le n} (a_{g_1}^i a_{g_2}^i + a_{g_1}^i a_{g_2}^j + a_{g_2}^i a_{g_1}^j + a_{g_1}^j a_{g_2}^j) =$$

$$n \sum_{j=1}^{n} a_{g_1}^j a_{g_2}^j + \sum_{1 \le i < j \le n} (a_{g_1}^i + a_{g_1}^j)(a_{g_2}^i + a_{g_2}^j)$$

Multiplication protocol

- Each party P_i engages in the two-party multiplication protocol with all other parties P_j.
- As result, party P_i learns the values $c^{i,j}$, such that

$$c^{i,j} + c^{j,i} = (a^i_{g_1} + a^i_{g_2}) \cdot (a^j_{g_1} + a^j_{g_2})$$

- P_i defines $a^i_g = n \cdot a^i_{g_1} a^i_{g_2} + \sum_{j \neq i} c^{i,j}$.

Exercise. Correctness, security? Uniformity of the values a^i_g?

Oblivious transfer in the malicious model

- Bellare-Micali construction (1-out-of-2 OT):
- Let G be a group with hard Diffie-Hellman problem. Let g generate G. Let $p = |G|$.
- Sender randomly picks $C \in G$ and sends it to the receiver.
- Receiver chooses $x \in \mathbb{Z}_p$ and defines $h_b = g^x$, $h_{1-b} = c/h_b$. Sends h_0, h_1 to the sender.
- Sender checks that $h_0 h_1 = C$. Uses ElGamal encryption to encrypt m_i with h_i. Sends

$$(g^{r_0}, m_0 \cdot h_0^{r_0}), (g^{r_1}, m_1 \cdot h_1^{r_1})$$

  to the receiver.
- Receiver decrypts the ciphertext that he can decrypt and learns m_b.

Exercise. Security?

Naor-Pinkas construction

Let G, g, p be as before.

- Receiver picks $s, t, c_{1-b} \in_R \mathbb{Z}_p$, defines $c_b = st \bmod p$, $x = g^s$, $y = g^t$, $z_i = g^{c_i}$. Sends x, y, z_0, z_1 to sender.
- Sender checks that $z_0 \neq z_1$. Picks random $r_0, r'_0, r_1, r'_1 \in \mathbb{Z}_p^*$ and returns to the receiver

$$((xg^{r_0})^{r'_0}, m_0 \cdot (z_0 y^{r_0})^{r'_0}), ((xg^{r_1})^{r'_1}, m_1 \cdot (z_1 y^{r_1})^{r'_1})$$

- The receiver...

Exercise. What is the receiver going to do with the values it got? What about security?

Exercise. Generalize this construction to 1-out-of-n OT.

Securing the original OT

Exercise. The original OT-protocol can also be secured by letting the receiver first commit to the randomness he's going to use, and then letting him prove in zero knowledge that he really used that randomness. Work out the details for 1-out-of-2 OT.