Oblivious Transfer & Yao's Garbled Circuits Protocols

C.V.H. SRI HARSHA
M. BHARGAV SRI VENKATESH

Oblivious Transfer (OT)

Setup:

- P1 (sender) has a set of N values $\{ s_1, s_2, \ldots.. s_N \}$.
- P2 wants to know particular i^{th} value of above set.

Goal:

Output nothing to P1, and i^{th} value of the set to P2 such that preventing P2 from learning any other value of the set.

Simple OT protocol

- 1-out-of-2 OT protocol.
- Receiver generates a public-private key pair and a random number indistinguishable from the generated public key.
- Sends k_0 , k_1 to the sender, k_i is public key (i^{th} value to be recovered) and k_{1-i} is the random number.

Simple OT protocol

- Sender sends encrypted values to the receiver.
- Receiver can decrypt only one of them, as private key is known only for one of them.
- Can be extended to 1-out-of-N OT. This works only in Semi-Honest setup.

Malicious-Secure OT protocol

Setup:

- P1 (sender) has a set of 2 strings { s_0, s_1} .
- P1 (sender) and P2 (receiver) select q and g such that g is a generator for Z^*_q.
- P1 selects a value C such that P2 does not know the discrete log of C in Z^*_q.

Malicious-Secure OT protocol

Setup:

- P2 selects i from {0, 1} corresponding to whether P2 wants s_0 or s_1.
- P2 also selects a random $0 \leq x_i < q - 1$.
- P2 sets $b_i = g^{x_i}$ and $b_{1-i} = C.g^{-x_i}$ where (b_0, b_1) and (i, x_i) form P1 public and private keys, respectively.

Malicious-Secure OT protocol

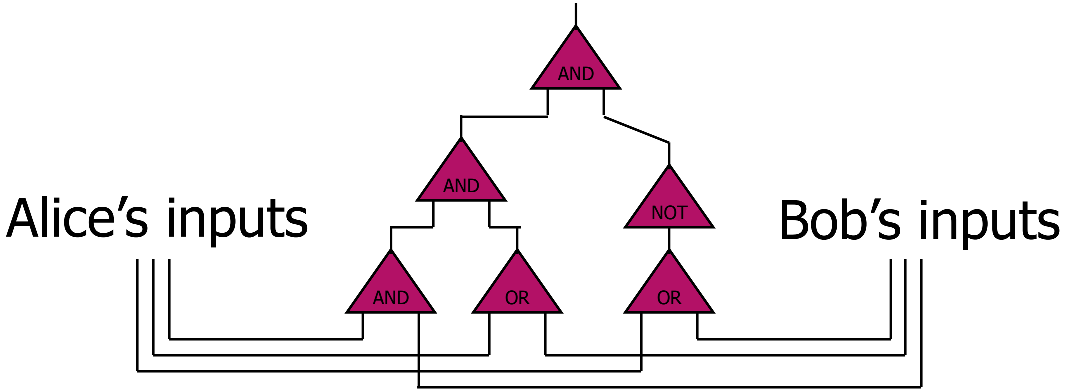
- P1 checks the validity of P2's public keys by verifying that $b_0 . b_1 = C$.
- If not, P1 aborts.
- P1 selects y_0, y_1 such that $0 \leq y_0, y_1 < q\text{-}1$.

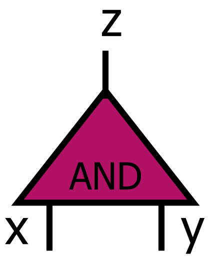
Malicious-Secure OT protocol

- P1 sends P2, $a_0 = g^{y_0}$ and $a_1 = g^{y_1}$.
- P1 also generates $z_0 = b_0^{y_0}$ and $z_1 = b_1^{y_1}$.
- P1 sends P2 $r_0 = s_0 \oplus z_0$, $r_1 = s_1 \oplus z_1$.
- P2 computes $z_i = a_i^{x_i}$ and then receives s_i by computing $s_i = r_i \oplus z_i$.

Yao’s protocol

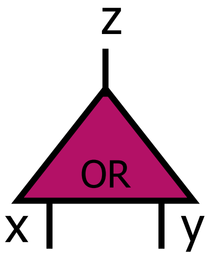
- Compute any function **securely** in the **semi-honest** model
- First, convert the function into a **Boolean circuit**





Truth table:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

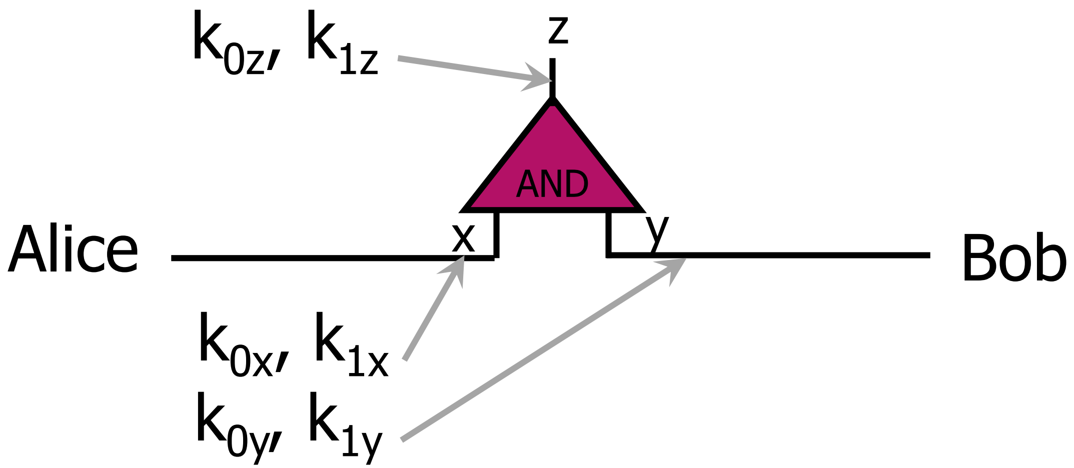


Truth table:

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

1: Pick Random Keys For Each Wire

- Next, evaluate **<u>one gate</u>** securely
  - Later, generalize to the entire circuit
- Alice picks two **random keys** for each wire
  - One key corresponds to “0”, the other to “1”
  - 6 keys in total for a gate with 2 input wires



2: Encrypt Truth Table

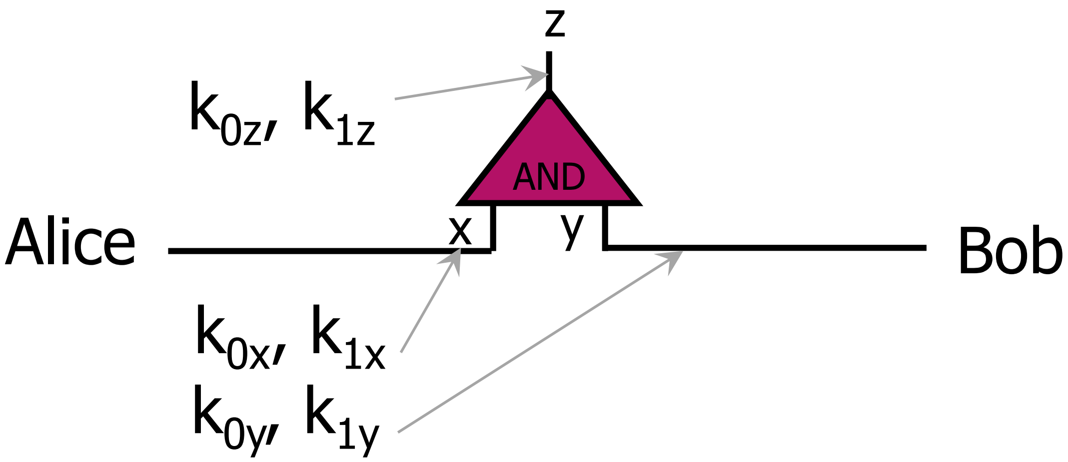
- Alice encrypts **each row** of the truth table by encrypting the output-wire key with the corresponding pair of input-wire keys

Original truth table:

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Encrypted truth table:

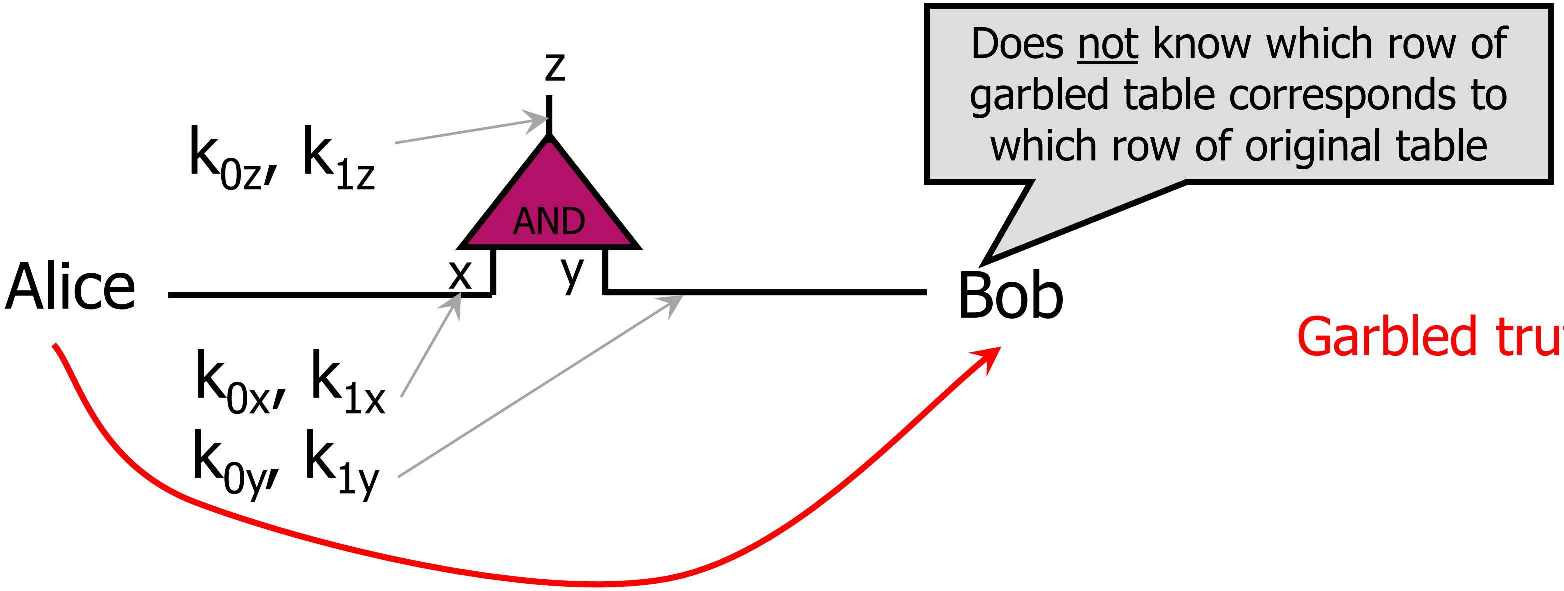
$$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$$
$$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$$
$$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$$
$$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$$



3: Send Garbled Truth Table

- Alice randomly permutes ("garbles") encrypted truth table and sends it to Bob

$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$



Garbled truth table:

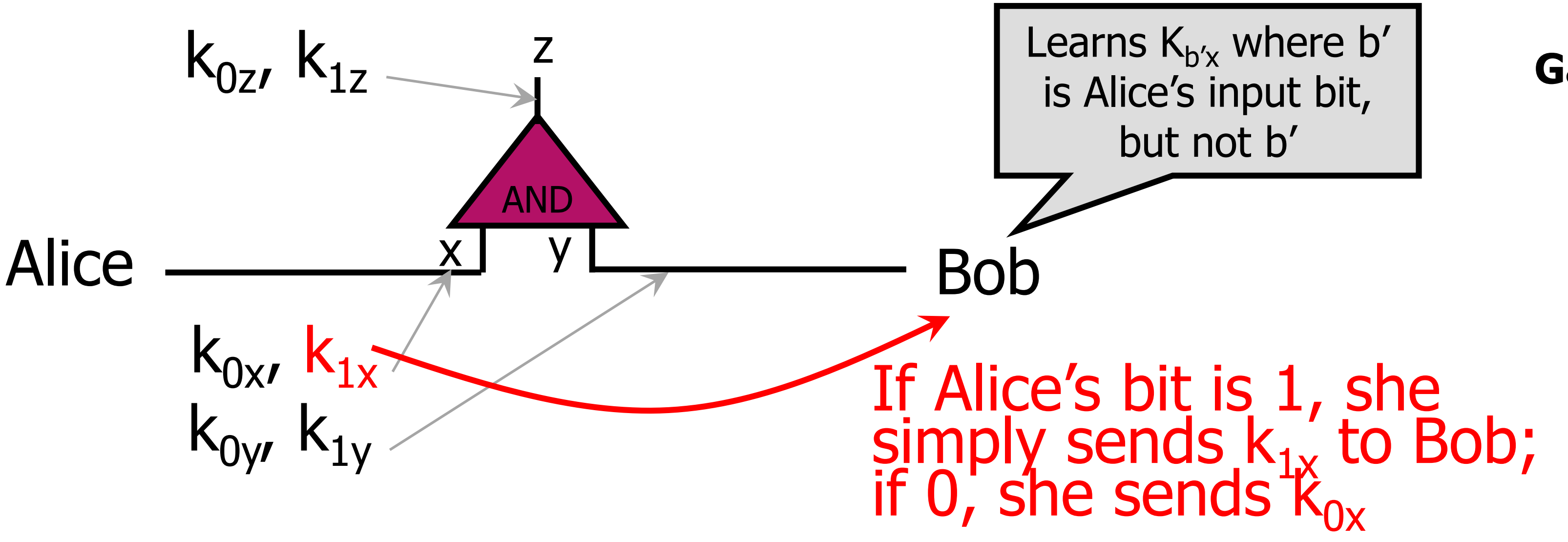
$E_{k_{1x}}(E_{k_{0y}}(k_{0z}))$
$E_{k_{0x}}(E_{k_{1y}}(k_{0z}))$
$E_{k_{1x}}(E_{k_{1y}}(k_{1z}))$
$E_{k_{0x}}(E_{k_{0y}}(k_{0z}))$

4: Send Keys For Alice's Inputs

- Alice sends the key corresponding to her input bit
  - Keys are random, so Bob does not learn what this bit is.

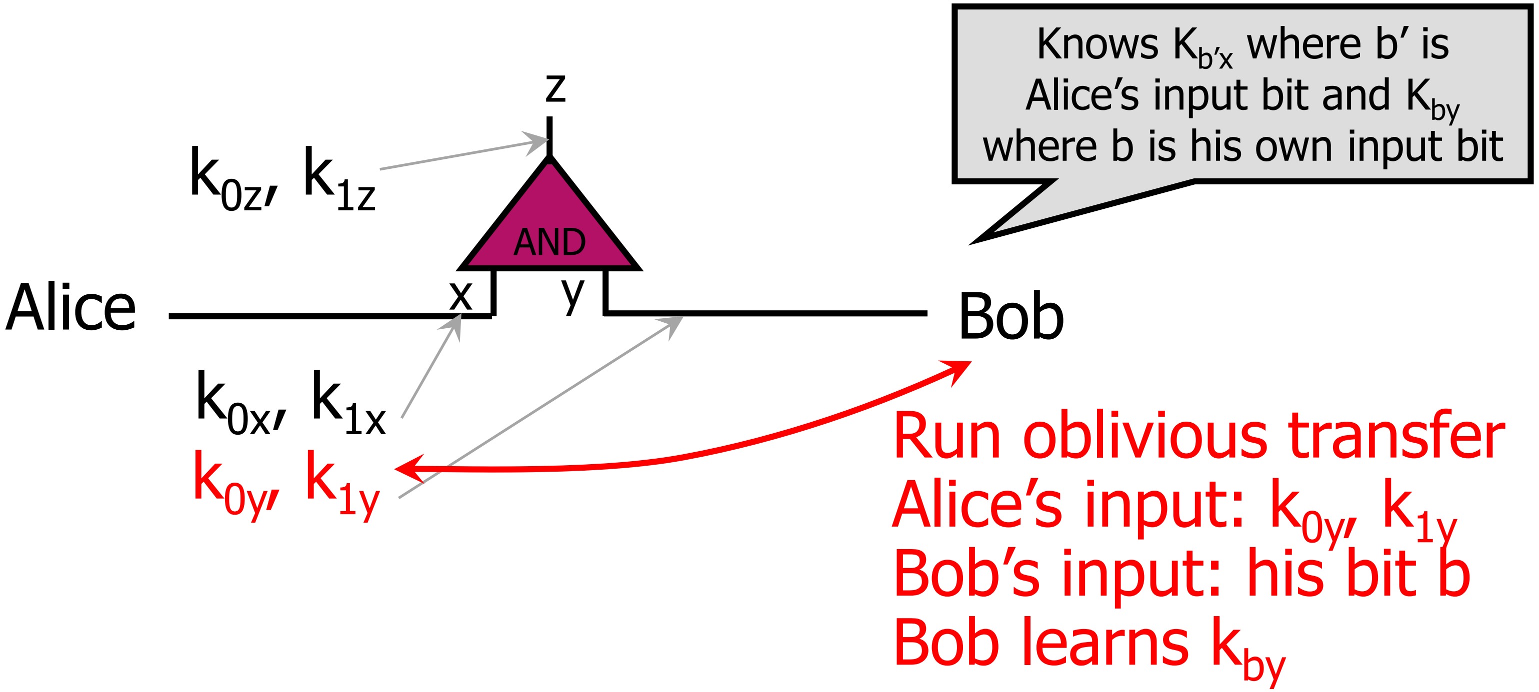
$E_{k1x}(E_{k0y}(k_{0z}))$
$E_{k0x}(E_{k1y}(k_{0z}))$
$E_{k1x}(E_{k1y}(k_{1z}))$
$E_{k0x}(E_{k0y}(k_{0z}))$

Garbled truth table



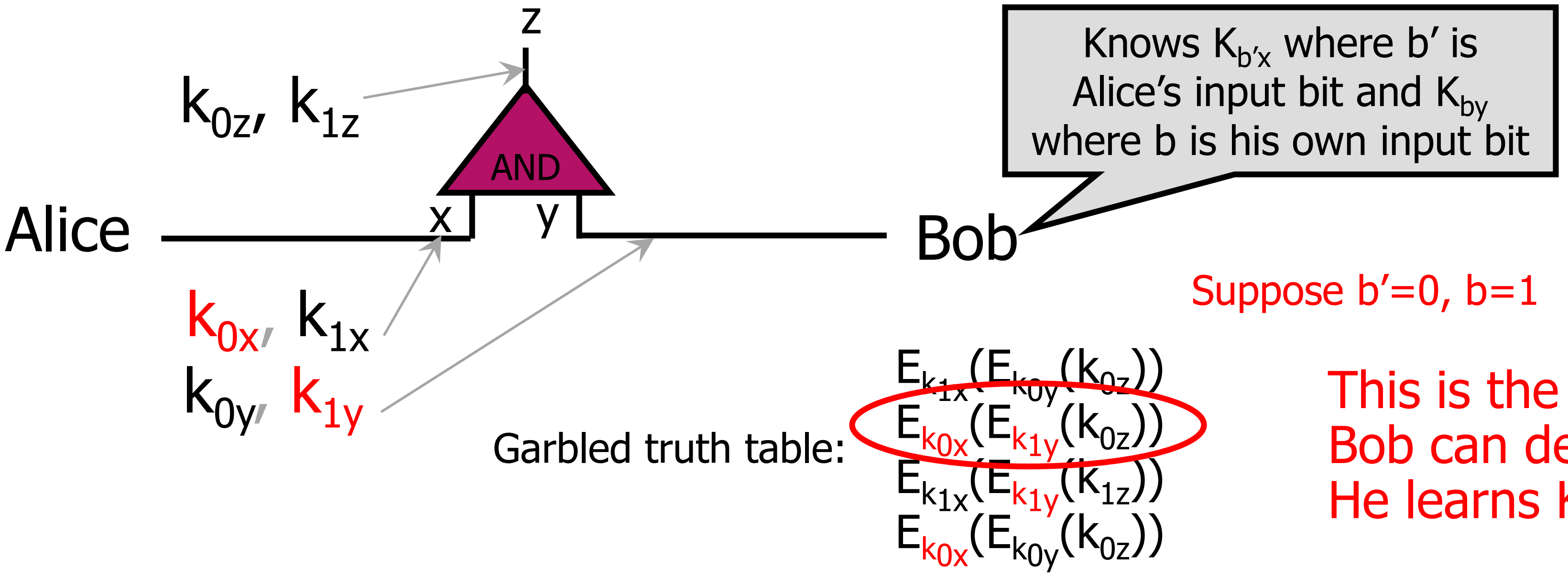
5: Use OT on Keys for Bob's Input

- Alice and Bob run **oblivious transfer protocol**
  - Alice's input is the two keys corresponding to Bob's wire
  - Bob's input into OT is simply his 1-bit input on that wire



6: Evaluate Garbled Gate

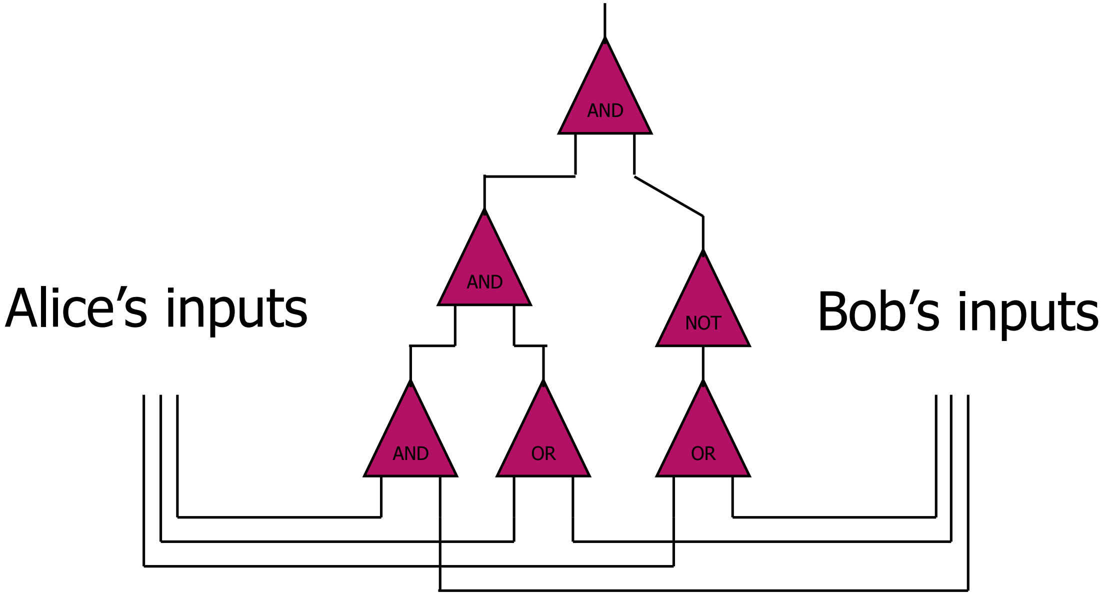
- Using the two keys that he learned, Bob decrypts exactly one of the output-wire keys and he does not learn if this key corresponds to 0 or 1



This is the only row Bob can decrypt.
He learns K_{0z}

7: Evaluate Entire Circuit

- In this way, Bob evaluates entire garbled circuit
  - For each wire in the circuit, Bob learns only one key
  - It corresponds to 0 or 1 (Bob does not know which)
    - Therefore, Bob does not learn intermediate values
- Bob tells Alice the key for the final output wire and she tells him if it corresponds to 0 or 1
  - Bob does not tell her intermediate wire keys



Drawback

- Above mentioned scheme can work only in a semi-honest setup.
- Receiver cannot see the correctness of circuit construction
- Sender can send incorrect garbled inputs to the receiver.

Securing Circuit Construction

- Standard Cut-and-Choose approach
- P1 constructs ‘m’ versions of the circuit, each structured identically but garbled differently so that the keys for each gate in each circuit are unique.
- Additionally, P1 generates a “commitment" for each of his garbled inputs, which for simplicity can be understood to be a simple hash of the inputs.

Cut-and-Choose

- P1 then sends each of these pairs of garbled circuits and associated input commitments to P2, who selects ‘m-1’ versions of the circuit to verify.

- P1 de-garbles each of the ‘m-1’ selected circuits, so that P2 can see the underlying circuit

- This reduces the chances of P1 tricking P2 into computing a corrupted circuit to 1/m .

Further Securing Cut-and-Choose

- Instead of P1 revealing 'm-1' circuits, P2 select only m/2 circuits to be revealed. P2 computes the remaining m/2 circuits and takes the majority result.

- P1 would only succeed in having P2 output a corrupt result if
  - 1. P1 constructs more than m/4 of the circuits corruptly, and
  - 2. None of the corrupt m/4 circuits are among the m/2 circuits P2 selected to be revealed.

Further Securing Cut-and-Choose

- P1's chance of success in such a scenario is $2^{-0.311m}$, where m is the number of circuits generated

- Still not secure against Corrupt Inputs.

Thank you

References

- Foundations of Cryptography, Volume II - Basic Applications, Oded Goldreich
- Yao's Garbled Circuits: Recent Directions and Implementations, Peter Snyder
- Privacy Preserving Data Mining, Yehuda Lindell, Benny Pinkas
- A. C. Yao, How to generate and exchange secrets, *Proceedings 27th Symposium on Foundations of Computer Science (FOCS)*, IEEE, 1986, pp. 162–167.
- CS 380S, Yao's Protocol, Vitaly Shmatikov