Yao's Garbled Circuit

Konstantinos Gkikas

October 23, 2014

What is Yao's construction?

A method that enables two parties with private inputs x and y to jointly compute a function $f(x, y)$.

What is Yao's construction?

A method that enables two parties with private inputs x and y to jointly compute a function $f(x, y)$.

Presented by Yao in a talk, can not be found in any of his papers.

What is Yao's construction?

A method that enables two parties with private inputs x and y to jointly compute a function $f(x, y)$.

Presented by Yao in a talk, can not be found in any of his papers.

Privacy Nothing is learned from the protocol other than the output.

What is Yao's construction?

A method that enables two parties with private inputs x and y to jointly compute a function $f(x, y)$.

Presented by Yao in a talk, can not be found in any of his papers.

Privacy Nothing is learned from the protocol other than the output.

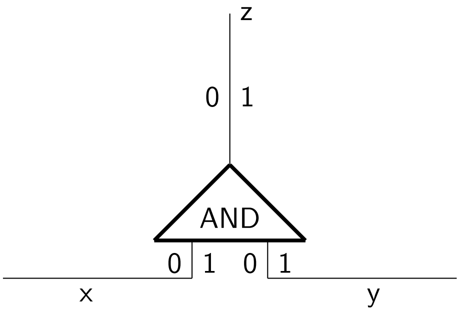
Based on a Boolean circuit.

How does it work?

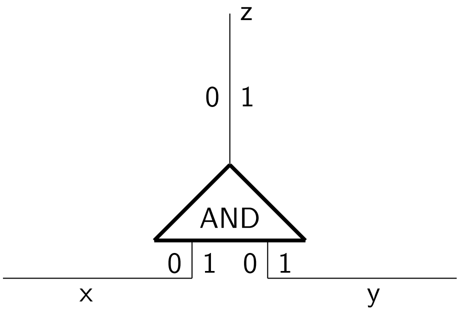
Transform the function $f(x, y)$ into a garbled boolean circuit $C(x, y)$.

- A garbled boolean circuit is a collection of garbled boolean gates.
- First understand what a single garbled gate is and then easily generalize to the entire garbled circuit.

Garbled Gate: What is that?

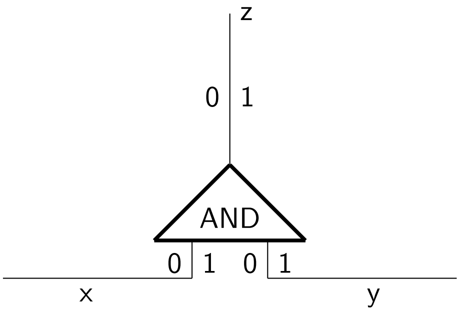


Garbled Gate: What is that?



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

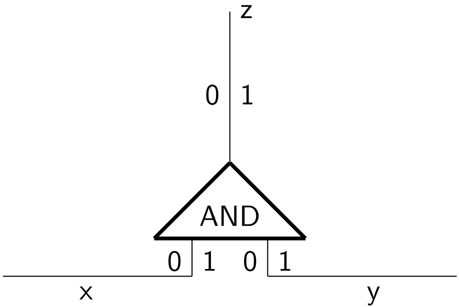
Garbled Gate: What is that?



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

- x,y are called INPUT wires, z is called OUTPUT wire

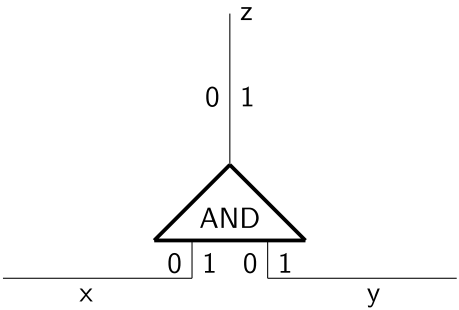
Garbled Gate: What is that?



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

- x,y are called INPUT wires, z is called OUTPUT wire
- For each of these three wires we have two values: $0, 1$ (input values for the input wires, output values for the output wire).

Garbled Gate: What is that?



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

- x,y are called INPUT wires, z is called OUTPUT wire
- For each of these three wires we have two values: $0, 1$ (input values for the input wires, output values for the output wire).
- Once input values a, b are selected, we want to compute $g(a, b)$ securely.

Construction of a Garbled Gate

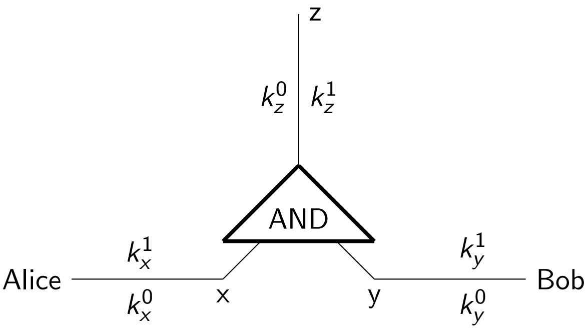
How do we do that?

Construction of a Garbled Gate

How do we do that?
For each wire x, y, z, specify two random values, corresponding to 0 and 1:

Construction of a Garbled Gate

How do we do that?
For each wire x, y, z, specify two random values, corresponding to 0 and 1:



- We need to "associate" k_z^0, k_z^1 with $k_x^0, k_x^1, k_y^0, k_y^1$. We will do that using the Garbled Computation Table (GCT).

Garbled Computation Table (GCT)

- View $k_z^0, k_z^1, k_x^0, k_x^1, k_y^0, k_y^1$ as encryption keys and encrypt k_z^0, k_z^1 under appropriate pairs of input keys.

Garbled Computation Table (GCT)

- View $k_z^0, k_z^1, k_x^0, k_x^1, k_y^0, k_y^1$ as encryption keys and encrypt k_z^0, k_z^1 under appropriate pairs of input keys.

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

input wire **x**	input wire **y**	output wire **z**	GCT
k_x^0	k_y^0	k_z^0	$E_{k_x^0}(E_{k_y^0}(k_z^0))$
k_x^0	k_y^1	k_z^0	$E_{k_x^0}(E_{k_y^1}(k_z^0))$
k_x^1	k_y^0	k_z^0	$E_{k_x^1}(E_{k_y^0}(k_z^0))$
k_x^1	k_y^1	k_z^1	$E_{k_x^1}(E_{k_y^1}(k_z^1))$

Garbled Computation Table (GCT)

- View $k_z^0, k_z^1, k_x^0, k_x^1, k_y^0, k_y^1$ as encryption keys and encrypt k_z^0, k_z^1 under appropriate pairs of input keys.

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

input wire **x**	input wire **y**	output wire **z**	GCT
k_x^0	k_y^0	k_z^0	$E_{k_x^0}(E_{k_y^0}(k_z^0))$
k_x^0	k_y^1	k_z^0	$E_{k_x^0}(E_{k_y^1}(k_z^0))$
k_x^1	k_y^0	k_z^0	$E_{k_x^1}(E_{k_y^0}(k_z^0))$
k_x^1	k_y^1	k_z^1	$E_{k_x^1}(E_{k_y^1}(k_z^1))$

- NOTE: Given two input keys k_x^a and k_y^b, only one row of the GCT can be decrypted correctly, namely: $E_{k_x^a}(E_{k_y^b}(k_z^{g(a,b)}))$.

Construction of a Garbled Gate

- Alice picks two random keys for each wire thus she has 6 keys in total.

Construction of a Garbled Gate

- Alice picks two random keys for each wire thus she has 6 keys in total.
- She encrypts each row of the table, creating the GCT:

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

Construction of a Garbled Gate

- Alice picks two random keys for each wire thus she has 6 keys in total.
- She encrypts each row of the table, creating the GCT:

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

- She permutes it (rearranges it), so that the key's position reveals nothing about the value that it is associated with.

Construction of a Garbled Gate

- Alice picks two random keys for each wire thus she has 6 keys in total.
- She encrypts each row of the table, creating the GCT:

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

- She permutes it (rearranges it), so that the key's position reveals nothing about the value that it is associated with.
- She sends it over to Bob, along with her input key $k_x^{b'}$, with b' her input value.

Construction of a Garbled Gate

- Alice picks two random keys for each wire thus she has 6 keys in total.
- She encrypts each row of the table, creating the GCT:

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

- She permutes it (rearranges it), so that the key's position reveals nothing about the value that it is associated with.
- She sends it over to Bob, along with her input key $k_x^{b'}$, with b' her input value.
- Bob still needs his own key to decrypt the GCT so Alice must send it to him.

Construction of a Garbled Gate

- Alice picks two random keys for each wire thus she has 6 keys in total.
- She encrypts each row of the table, creating the GCT:

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

- She permutes it (rearranges it), so that the key's position reveals nothing about the value that it is associated with.
- She sends it over to Bob, along with her input key $k_x^{b'}$, with b' her input value.
- Bob still needs his own key to decrypt the GCT so Alice must send it to him.
- If Alice sends both k_y^0, k_y^1 to Bob, then Bob can decrypt more and if Bob asks for Alice the key that corresponds to his input, then Alice learns Bob's input.

Oblivious Transfer

So how does Bob receive his key?
By using Oblivious Transfer.

Oblivious Transfer

So how does Bob receive his key?
By using Oblivious Transfer. A protocol in which:

- sender inputs x_0, x_1
- receiver inputs s
- receiver obtains x_s

- Sender learns nothing about s, receiver learns nothing about x_{s-1}.

Computation of the garbled gate

Bob now has $k_x^{b'}, k_y^{b}$ and the GCT and he can compute the gate by decrypting GCT.

Computation of the garbled gate

Bob now has $k_x^{b'}, k_y^b$ and the GCT and he can compute the gate by decrypting GCT.
He can decrypt only one line of the GCT, exactly because of its construction.

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

Computation of the garbled gate

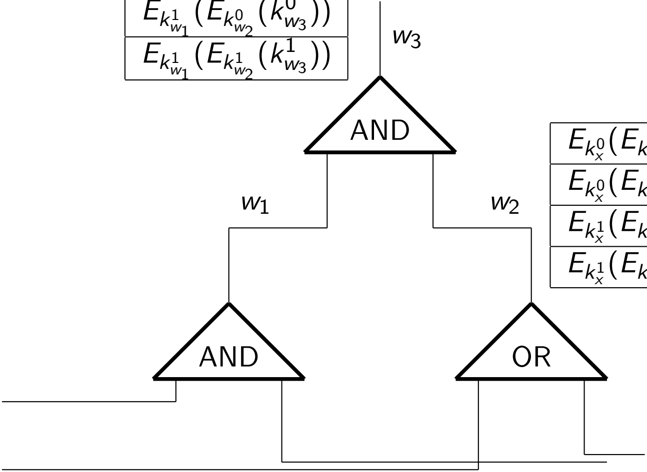
Bob now has $k_x^{b'}, k_y^{b}$ and the GCT and he can compute the gate by decrypting GCT.
He can decrypt only one line of the GCT, exactly because of its construction.

$E_{k_x^0}(E_{k_y^0}(k_z^0))$
$E_{k_x^0}(E_{k_y^1}(k_z^0))$
$E_{k_x^1}(E_{k_y^0}(k_z^0))$
$E_{k_x^1}(E_{k_y^1}(k_z^1))$

Sends output $k_z^{g(b',b)}$ to Alice and the computation of the garbled gate is over.

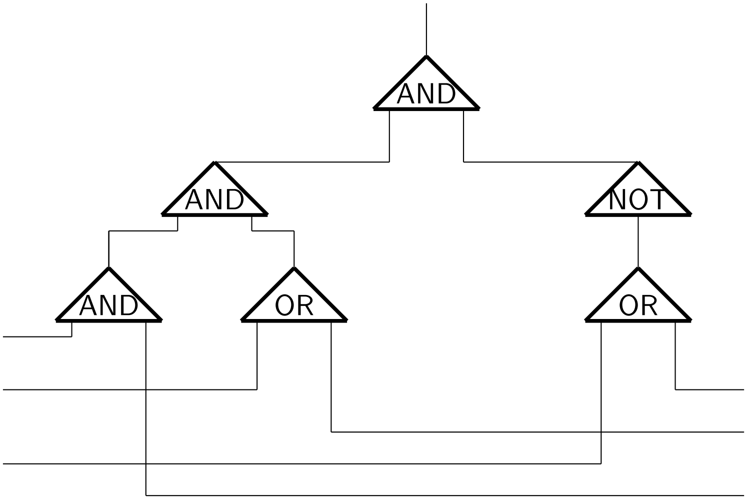
Generalizing to a circuit

$E_{k^0_{w_1}}(E_{k^0_{w_2}}(k^0_{w_3}))$
$E_{k^0_{w_1}}(E_{k^1_{w_2}}(k^0_{w_3}))$
$E_{k^1_{w_1}}(E_{k^0_{w_2}}(k^0_{w_3}))$
$E_{k^1_{w_1}}(E_{k^1_{w_2}}(k^1_{w_3}))$



$E_{k^0_x}(E_{k^0_y}(k^0_z))$
$E_{k^0_x}(E_{k^1_y}(k^1_z))$
$E_{k^1_x}(E_{k^0_y}(k^1_z))$
$E_{k^1_x}(E_{k^1_y}(k^1_z))$

Generalizing to a circuit



Outline of the Protocol

- P_1 constructs a garbled circuit.

Outline of the Protocol

- P_1 constructs a garbled circuit.
- P_1 sends to P_2 the keys associated with its inputs.

Outline of the Protocol

- P_1 constructs a garbled circuit.
- P_1 sends to P_2 the keys associated with its inputs.
- P_1 and P_2 use oblivious transfer so that:

Outline of the Protocol

- P_1 constructs a garbled circuit.
- P_1 sends to P_2 the keys associated with its inputs.
- P_1 and P_2 use oblivious transfer so that:
    - P_2 obtains the key associated with its input on all of its wires

Outline of the Protocol

- P_1 constructs a garbled circuit.
- P_1 sends to P_2 the keys associated with its inputs.
- P_1 and P_2 use oblivious transfer so that:
  - P_2 obtains the key associated with its input on all of its wires
  - P_1 learns nothing about P_2 inputs.

Outline of the Protocol

- P_1 constructs a garbled circuit.
- P_1 sends to P_2 the keys associated with its inputs.
- P_1 and P_2 use oblivious transfer so that:
  - P_2 obtains the key associated with its input on all of its wires
  - P_1 learns nothing about P_2 inputs.

P_2 computes the circuit and sends the output back to P_1.

Simplifying the above

- Two padlock keys for each wire.
- 4 doubly-locked boxes for each gate, such that:
  - locked in a way that a pair of keys can open only one
  - in each box there is a new key
- Open boxes one at a time until final output.
- No other information revealed (keys can not be associated with 0 or 1).
- Actual Construction: Replace each box with a row in GCT and physical keys with encryption keys

Proof of Security

- Yao's protocol is secure in the presence of semi-honest adversaries: follow the protocol, but attempt to learn information by analyzing the transcript of messages during the execution.

Proof of Security

- Yao's protocol is secure in the presence of semi-honest adversaries: follow the protocol, but attempt to learn information by analyzing the transcript of messages during the execution.
- Took 20 years to provide a full formal proof.
- Formal proof definitely not trivial.
- Proof can be found in the paper posted in the website of the course under Yao's garbled circuits and there is a proper citation of it in my handout.

Interesting Question

How does Bob know when he has decrypted correctly?
We use serial encryption, meaning that if Bob decrypts the GCT, he will obtain 4 random values. How can he distinguish which one of these is the correct one?
Since what Bob sees are encryption keys, not associated with output values, he seems to be unable to decide which key is the correct one.

Interesting Question

How does Bob know when he has decrypted correctly?
We use serial encryption, meaning that if Bob decrypts the GCT, he will obtain 4 random values. How can he distinguish which one of these is the correct one?
Since what Bob sees are encryption keys, not associated with output values, he seems to be unable to decide which key is the correct one.
There are many ways in order to fix this issue. One of them is the following:

Interesting Question

How does Bob know when he has decrypted correctly?
We use serial encryption, meaning that if Bob decrypts the GCT, he will obtain 4 random values. How can he distinguish which one of these is the correct one?
Since what Bob sees are encryption keys, not associated with output values, he seems to be unable to decide which key is the correct one.
There are many ways in order to fix this issue. One of them is the following:

- Use encryption based on a PRF and use redundant zeros while encrypting.
- In this way, only the correct keys give a redundant block.

Interesting Question

How does Bob know when he has decrypted correctly?

We use serial encryption, meaning that if Bob decrypts the GCT, he will obtain 4 random values. How can he distinguish which one of these is the correct one?

Since what Bob sees are encryption keys, not associated with output values, he seems to be unable to decide which key is the correct one.

There are many ways in order to fix this issue. One of them is the following:

- Use encryption based on a PRF and use redundant zeros while encrypting.
- In this way, only the correct keys give a redundant block.
- For instance while encrypting the key, also encrypt 100 zeros.
- Then, the correct value is followed by 100 zeros.
- Probability that an incorrect value is followed by 100 zeros is negligible.