Hand out for Presentation: Yao's garbled circuits

KONSTANTINOS GKIKAS

October 24, 2014

- Yao's garbled circuit is a method that enables two parties to jointly compute a function securely (no information other than the output of the function is revealed) in the presence of semi-honest adversaries. It took 20 years to formally prove this protocol's security and the formal proof can be found in [1] .
- Note: For what follows, I assume that the reader knows what a boolean circuit and a boolean gate is. Otherwise, he can always refer to the slides of my presentation.

The key idea underlying Yao's construction is to transform the function f into a boolean circuit.
We do that by associating two random values to each wire of the circuit, one representing value 0 and the other representing value 1.
The idea is that the circuit is computed gate-by-gate until the very final output of the circuit is computed and shared by the two parties.
Construction and Computation of a single garbled gate.
Construction
Essentially, a garbled gate is the following table:

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

- Values k_x^a, k_y^b, $a, b \in \{0, 1\}$ represent Party's 1 and Party's 2 inputs respectively. On the other hand, k_z^c, $c \in \{0, 1\}$ represent the corresponding output-wire values (x, y are considered to be input wires, while z is considered to be the output wire).
- Viewing k_x^a, k_y^b, k_z^c as encryption keys, we doubly-encrypt the output wire keys under the pairs of corresponding input wire keys.
- Note that this "correspondence" is determined by the type of the gate that we have. The above represent an AND gate. Considering the truth table of the AND operator these "correspondences" become obvious.

Computation
Now the above table is computed, by simply decrypting the doubly-encrypted values under the corresponding pair of input values.

- NOTE: Only one row of the above table can be decrypted correctly, therefore ensuring that only one output value is obtained for any given pair of input values.

- Moreover, obtaining the output key k_z^c reveals nothing about the value of c.

Garbled Gate "in action"
Assume that as always our two parties are Alice and Bob. We will see how Alice constructs the garbled gate and furthermore how the output is computed.

- Alice generates 6 random values: $k_x^0, k_x^1, k_y^0, k_y^1, k_z^0, k_z^1$ Let x be Alice's input wire, y Bob's and z the output-wire.
- Alice selects her input and sends to Bob her key associated with her input: $k_x^{b'}$, $b' \in \{0, 1\}$.
- Alice double-encrypts the output wire keys under the appropriate pairs of input keys, constructing the garbled computation table, she permutes it (rearranges it) and sends it over to Bob.
- Bob still need his own key in order to decrypt the table.
- He gets it using oblivious transfer.
  - Oblivious transfer is a protocol in which sender inputs two values x_0, x_1 and the receiver inputs a bit $\sigma \in \{0, 1\}$. The receiver obtains x_σ while sender learns nothing about σ and receiver learns nothing about $x_{\sigma-1}$.
- Now Bob has everything he needs to decrypt the table, obtaining one correct output and shares it with Alice.

Generalizing
Now a whole garbled circuit is computed gate-by-gate. For each of the gates, Bob receives Alice's keys and a garbled table and also his own key using Oblivious Transfer. Computing each gate and using the outputs he receives as input for the next gate he moves on to obtain the final output of the circuit.
It is important that Bob does not share any intermediate values with Alice while computing the circuit, otherwise Alice would learn more than what she is supposed to know.

References

[1] Yehuda Lindell and Benny Pinkas. A proof of security of yaos protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[2] Youtube. The yao construction and its proof of security, 2011.

[1] [2]