Secure Computation - II

CS 601.642/442 Modern Cryptography

Fall 2018

Securely Computing *any* Function

Main question: How can Alice and Bob securely compute *any* function f over their private inputs x and y?

Solution: Using Yao's *garbled circuits* with OT

Garbled Circuits

A Garbling Scheme consists of two procedures $(\mathsf{Garble}, \mathsf{Eval})$:

- $\mathsf{Garble}(C)$: Takes as input a circuit C and outputs a collection of garbled gates $\hat{\mathsf{G}}$ and garbled input wires $\hat{\mathsf{In}}$ where

$$\hat{\mathsf{G}} = \{\hat{g}_1, \ldots, \hat{g}_{|C|}\},$$

$$\hat{\mathsf{In}} = \{\hat{\mathsf{in}}_1, \ldots, \hat{\mathsf{in}}_n\}.$$

- $\mathsf{Eval}(\hat{\mathsf{G}}, \hat{\mathsf{In}}_x)$: Takes as input a garbled circuit $\hat{\mathsf{G}}$ and garbled input wires $\hat{\mathsf{In}}_x$ corresponding to an input x and outputs $z = C(x)$

Garbled Circuits: Ideas

- Each wire i in the circuit C is associated with two keys (k_0^i, k_1^i) of a secret-key encryption scheme, one corresponding to the wire value being 0 and other for wire value being 1
- For an input x, the evaluator is given the input wire keys $(k_{x_1}^1, \ldots, k_{x_n}^n)$ corresponding to x. Furthermore, for every gate g in C, it is also given an "encrypted" truth table of g
- We want the evaluator to use the input wire keys and the encrypted truth tables to "uncover" a single key k_v^i for every internal wire i corresponding to the value v of that wire. However, k_{1-v}^i should remain hidden from the evaluator

Special Encryption Scheme

Special Encryption Scheme: We need a secret-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with an extra property: there exists a negligible function $\nu(\cdot)$ s.t. for every n and every message $m \in \{0,1\}^n$,

$$\Pr[k \leftarrow \mathsf{Gen}(1^n), k' \leftarrow \mathsf{Gen}(1^n), \mathsf{Dec}_{k'}(\mathsf{Enc}_k(m)) = \perp] > 1 - \nu(n)$$

That is, if a ciphertext is decrypted using the "wrong" key, then the answer is always $\perp$

Construction: Modify the secret-key encryption scheme discussed earlier in the class s.t. instead of encrypting m, we encrypt $0^n \| m$. Upon decrypting, check if the first n bits of the message are all 0's; if not, then output $\perp$.

Garbled Circuits: Construction

Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a special encryption scheme. Assign an index to each wire in C s.t. the input wires have indices $1, \ldots, n$.

$\mathsf{Garble}(C)$:

- For every non-output wire i in C, sample $k_0^i \leftarrow \mathsf{Gen}(1^n)$, $k_1^i \leftarrow \mathsf{Gen}(1^n)$. For every output wire i in C, set $k_0^i = 0$, $k_1^i = 1$.
- For every $i \in [n]$, set $\mathsf{in}_i = (k_0^i, k_1^i)$. Set $\mathsf{In} = (\mathsf{in}_1, \ldots, \mathsf{in}_n)$
- For every gate g in C with input wires (i, j), output wire ℓ:

First Input	Second Input	Output
k_0^i	k_0^j	$z_1 = \mathsf{Enc}_{k_0^i}(\mathsf{Enc}_{k_0^j}(k_{g(0,0)}^\ell))$
k_0^i	k_1^j	$z_2 = \mathsf{Enc}_{k_0^i}(\mathsf{Enc}_{k_1^j}(k_{g(0,1)}^\ell))$
k_1^i	k_0^j	$z_3 = \mathsf{Enc}_{k_1^i}(\mathsf{Enc}_{k_0^j}(k_{g(1,0)}^\ell))$
k_1^i	k_1^j	$z_4 = \mathsf{Enc}_{k_1^i}(\mathsf{Enc}_{k_1^j}(k_{g(1,1)}^\ell))$

Set $\hat{g} = \mathsf{RandomShuffle}(z_1, z_2, z_3, z_4)$. Output $(\hat{\mathsf{G}} = (\hat{g}_1, \ldots, \hat{g}_{|C|}), \hat{\mathsf{In}})$

Garbled Circuits: Construction (contd.)

Think: Why is RandomShuffle necessary?

$\mathsf{Eval}(\hat{\mathsf{G}}, \hat{\mathsf{In}}_x)$:

- Parse $\hat{\mathsf{G}} = (\hat{g}_1, \ldots, \hat{g}_{|C|})$, $\hat{\mathsf{In}}_x = (k^1, \ldots, k^n)$
- Parse $\hat{g}_i = (\hat{g}_i^1, \ldots, \hat{g}_i^4)$
- Decrypt each garbled gate $\hat{g}_i$ one-by-one, in a canonical order:
  - Let k^i and k^j be the input wire keys for gate g.
  - Repeat the following for every $p \in [4]$:

$$\alpha_p = \mathsf{Dec}_{k^i}(\mathsf{Dec}_{k^j}(\hat{g}_i^p))$$

  If $\exists \alpha_p \neq \perp$, set $k^\ell = \alpha_p$

- Let out_i be the value obtained for each output wire i. Output $\mathsf{out} = (\mathsf{out}_1, \ldots, \mathsf{out}_n)$

Secure Computation from Garbled Circuits

A plausible strategy for computing $C(x, y)$ using Garbled Circuits:

- A generates a garbled circuit for $C(\cdot, \cdot)$ along with garbled wire keys for first and second input to C
- A sends the garbled wire keys corresponding to its input x along with the garbled circuit to B
- However, in order to evaluate the garbled circuit on (x, y), B also needs the garbled wire keys corresponding to its input y
- **Possible Solution:** A sends all the wire keys corresponding to the second input of C to B
- **Problem:** In this case, B can not only compute $C(x, y)$ but also $C(x, y')$ for any y' of its choice!
- **Solution:** A will transmit the garbled wire keys corresponding to B's input using Oblivious Transfer!

Secure Computation from Garbled Circuits: Details

Ingredients: Garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$, 1-out-of-2 OT scheme $\mathsf{OT} = (S, R)$

Common Input: Circuit C for $f(\cdot, \cdot)$

A**'s input:** $x = x_1, \ldots, x_n$, B**'s input:** $y = y_1, \ldots, y_n$

Protocol $\Pi = (A, B)$**:**

- $A \to B$: A computes $(\hat{G}, \hat{\mathsf{In}}) \leftarrow \mathsf{Garble}(C)$. Parse $\hat{\mathsf{In}} = (\hat{\mathsf{in}}_1, \ldots, \hat{\mathsf{in}}_{2n})$ where $\hat{\mathsf{in}}_i = (k_0^i, k_1^i)$. Set $\hat{\mathsf{In}}_x = (k_{x_1}^1, \ldots, k_{x_n}^n)$. Send $(\hat{\mathsf{G}}, \hat{\mathsf{In}}_x)$ to B.
- $A \leftrightarrow B$: For every $i \in [n]$, A and B run $\mathsf{OT} = (S, R)$ where A plays sender S with input (k_0^{n+i}, k_1^{n+i}) and B plays receiver R with input y_i. Let $\hat{\mathsf{In}}_y = (k_{y_1}^{n+1}, \ldots, k_{y_n}^{2n})$ be the outputs of the n OT executions received by B.
- B: B outputs $\mathsf{Eval}(\hat{\mathsf{G}}, \hat{\mathsf{In}}_x, \hat{\mathsf{In}}_y)$

Intuition for Security

Property 1: For every wire i, B only learns one of the two wire keys:

- **Input wires:** For input wires corresponding to A's input, it follows from protocol description. For input wires corresponding to B's input, it follows from security of OT
- **Internal Wires:** Follows from the security of the encryption scheme

Property 2: B does not know whether the key corresponds to wire value being 0 or 1 (except the keys corresponding to its own input wires).

Overall, B only learns the output and nothing else. A does not learn anything (in particular, B's input remains hidden from A due to security of OT)

Additional Reading: Read security proof from [Lindell-Pinkas'04]