Secure Computation - II

CS 601.642/442 Modern Cryptography

Fall 2018
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
A Garbling Scheme consists of two procedures (Garble, Eval):

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

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

  $$\hat{\ln} = \{\hat{\ln}_1, \ldots, \hat{\ln}_n\}.$$

- **Eval**($\hat{G}, \hat{\ln}_x$): Takes as input a garbled circuit $\hat{G}$ and garbled input wires $\hat{\ln}_x$ 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 $(k^i_0, k^i_1)$ 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 $(k^1_{x_1}, \ldots, k^n_{x_n})$ 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 $k^i_v$ for every internal wire $i$ corresponding to the value $v$ of that wire. However, $k^i_{1-v}$ should remain hidden from the evaluator.
Special Encryption Scheme: We need a secret-key encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) with an extra property: there exists a negligible function \(\nu(\cdot)\) s.t. for every \(n\) and every message \(m \in \{0, 1\}^n\),

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

That is, if a ciphertext is decrypted using the “wrong” key, then the answer is always \(\bot\)

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 \(\bot\).
Garbled Circuits: Construction

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

\textbf{Garble}(\(C\)):

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

<table>
<thead>
<tr>
<th>First Input</th>
<th>Second Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(k_0^i)</td>
<td>(k_0^j)</td>
<td>(z_1 = \text{Enc}<em>{k_0^i}(\text{Enc}</em>{k_0^j}(k_\ell^{g(0,0)})))</td>
</tr>
<tr>
<td>(k_0^i)</td>
<td>(k_1^j)</td>
<td>(z_2 = \text{Enc}<em>{k_0^i}(\text{Enc}</em>{k_1^j}(k_\ell^{g(0,1)})))</td>
</tr>
<tr>
<td>(k_1^i)</td>
<td>(k_0^j)</td>
<td>(z_3 = \text{Enc}<em>{k_1^i}(\text{Enc}</em>{k_0^j}(k_\ell^{g(1,0)})))</td>
</tr>
<tr>
<td>(k_1^i)</td>
<td>(k_1^j)</td>
<td>(z_4 = \text{Enc}<em>{k_1^i}(\text{Enc}</em>{k_1^j}(k_\ell^{g(1,1)})))</td>
</tr>
</tbody>
</table>

Set \(\hat{g} = \text{RandomShuffle}(z_1, z_2, z_3, z_4)\). Output \((\hat{G} = (\hat{g}_1, \ldots, \hat{g}_{|C|}), \hat{\text{In}})\).
Think: Why is RandomShuffle necessary?

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

- Parse $\hat{G} = (\hat{g}_1, \ldots, \hat{g}_{|\mathcal{C}|})$, $\hat{\ln}_x = (k^1, \ldots, k^n)$
- Parse $\hat{g}_i = (\hat{g}^{1}_i, \ldots, \hat{g}^{4}_i)$
- 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 $p \in [4]$:

$$\alpha_p = \text{Dec}_{k^i}(\text{Dec}_{k^j}(\hat{g}^p_i))$$

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

- Let $\text{out}_i$ be the value obtained for each output wire $i$. Output $\text{out} = (\text{out}_1, \ldots, \text{out}_n)$
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!
**Ingredients:** Garbling scheme \((\text{Garble}, \text{Eval})\), 1-out-of-2 OT scheme \(\text{OT} = (S, R)\)

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

\(A \rightarrow B\): \(A\) computes \((\hat{G}, \hat{In}) \leftarrow \text{Garble}(C)\). Parse \(\hat{In} = (\hat{in}_1, \ldots, \hat{in}_{2n})\) where \(\hat{in}_i = (k_{i0}^0, k_{i1}^0)\). Set \(\hat{In}_x = (k_{x1}^1, \ldots, k_{xn}^n)\). Send \((\hat{G}, \hat{In}_x)\) to \(B\).

\(A \leftrightarrow B\): For every \(i \in [n]\), \(A\) and \(B\) run \(\text{OT} = (S, R)\) 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 \(\hat{In}_y = (k_{y1}^{n+1}, \ldots, k_{yn}^{2n})\) be the outputs of the \(n\) OT executions received by \(B\).

\(B\): \(B\) outputs \(\text{Eval}(\hat{G}, \hat{In}_x, \hat{In}_y)\)
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]