Stacked Garbling Gadget

Vlad Kolesnikov
Georgia Tech

NIST MPTS 2023
Outline

• Garbled Circuits (GC)
• Gadget: Stacked Garbling (SGC)
• Applications and standardization comments
Functions are circuits

\[ F(x, y) \rightarrow \]

OR

AND
GC intuition: computing on encrypted values
GC intuition: computing on encrypted values

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>a^b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
GC intuition: computing on encrypted values

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a∧b</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
GC intuition: decoding encrypted output
GC intuition: OT for transferring input labels
Stacked garbling [HK20]

• Sequence of works [K18,HK20a,HK20b,HK21]
• Let’s question the circuit model of computation.
• But not too much..
• Just consider circuits with conditionals

Let C0, C1 be two arbitrary circuits. The space of circuits is defined as follows:

\[ C ::= \text{Netlist}(\cdot) \mid \text{Cond}(C0, C1) \mid \text{Seq}(C0, C1) \]
Stacked garbling [HK20]

\[ C ::= \text{Netlist}(\cdot) | \text{Cond}(C_0, C_1) | \text{Seq}(C_0, C_1) \]

HK20: Can evaluate \( \text{Cond}(C_0, C_1) \) while transmitting only one branch

Idea:

* the same GC material \( M \) is used for evaluation of \( C_0 \) and \( C_1 \).
* GC outputs a key to Eval which converts material \( M \) to a valid GC when evaluated on the active branch or to a random-looking string otherwise (Eval can’t distinguish)
* Eval evaluates both \( C_0, C_1 \). One of them will produce garbage labels. They are canceled (garbage-collected) by gadgets constructed by Garbler.
* **Material reuse** (novel general idea; works for other protocols as well)
Stacked garbling [HK20]
Stacked garbling [HK20]

For each branch, if it is active, Bob gets a good output label, otherwise he gets garbage output label. He can’t tell which is which (requires that GC material and labels look random – achieved by half-gates scheme).

Guess active branch 1
Expand seed as branch 2
Guess active branch 2
Expand seed as branch 1
Stacked garbling [HK20]

For active branch, Bob gets a valid label, otherwise he gets garbage output label.

We need to obliviously discard garbage.
Key idea: Bob is deterministic and Alice can emulate him and predict the possible garbage keys
Then Alice constructs a MUX gadget which collects garbage.
For each of $b$ branches, $E$ will attempt $b$ guesses.

Each produces different output wire keys (total $O(b^2)$).

To proceed past the conditional we must collect garbage outputs that result from each possible bad evaluation.

Garbler’s work is $O(b^2)$. 

**SGC**

- For each of $b$ branches, $E$ will attempt $b$ guesses.
- Each produces different output wire keys (total $O(b^2)$).
- To proceed past the conditional we must collect garbage outputs that result from each possible bad evaluation.
- Garbler’s work is $O(b^2)$. 

**Logstack**
Logstack (HK21)

Idea

- Consolidate garbage collection by setting branches/seeds into a binary tree
- Each branch’s garbage depends on which sibling subtree holds the active branch. Hence, each branch has ⌈log_2 b⌉ possible garbage labels
- G can precompute all garbage in $O(⌈\log b⌉)$ time and build a garbage-collecting multiplexer
Applications to threshold cryptography
From circuits to RAM machines

Need: compute F
Represent F as circuit vs as a C program
Implementing CPU with Stacked garbling

\[ \text{[HK20]} \text{ For circuit } C = Cond(C_0, C_1, \ldots, C_{n-1}) \]
Performance improvement factor \( n \)

**CPU is such a conditional circuit!**

Implement N CPU steps as sequence of N circuits. Each circuit ALU is now as large as a single instruction!
GC is basic

• It is a simple object; it is not a protocol
• Standardizing just GC gives cryptographic object with clean security properties.
• Optional OT/GC usage standardization makes is a secure MPC standard
• In MPTS’20, GC world was relatively simple. Since then there were some nice developments.
GC standardization

Basic GC is very stable.
Standardizing basic GC

• Not likely to hinder future algorithmic enhancements
• Encourage development and standardization of gadgets
• Will aid in Threshold crypto (mandate of this group),
  • and be a catalyst for MPC development and adoption.

So let’s go!