FPGA Implementation of Generator and Evaluator for Garbled Circuits

Xin Fang, Stratis Ioannidis, Miriam Leeser
Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

Abstract

Garbled Circuits (GC) is a protocol for implementing Secure Function Evaluation (SFE) which can evaluate any function and obtain the result while keeping each party’s input private. Recent advances have lead to a surge of GC implementations and applications to secure evaluation of a variety of different tasks. Several current implementations of garbled circuits are implemented as software. We show the feasibility of implementing garbled circuits on an FPGA and demonstrate that using parallelism significantly reduces the computational cost of evaluating AND gates of a GC over an FPGA. Compared with the existing platform ObliVM*, our implementation achieves 4.5*10^4 times faster.

Secure Function Evaluation

Given number of participants, P1, P2, ..., Pn, each have private data, respectively D1, D2, ..., Dn. Participants want to compute the value of a public function on that private data: f(D1, D2, ..., Dn) while keeping their own inputs secret.

Threat Model

Garbled circuits provide security guarantees under the semi-honest (a.k.a. honest but curious) threat model. In other words, all participants follow the protocols proposed as prescribed; however, these interested parties may elect to analyze protocol transcripts, even off-line, in order to infer some additional information.

Garbled Circuit Algorithm

Generating A Boolean Circuit Representation of the Function

Garbling Truth Tables

<table>
<thead>
<tr>
<th>w0</th>
<th>w1</th>
<th>w2</th>
<th>w3</th>
<th>w4</th>
<th>k1</th>
<th>k2</th>
<th>garbled value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>w0</td>
<td>k1</td>
<td>k2</td>
<td>H(k1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>w0</td>
<td>k1</td>
<td>k2</td>
<td>H(k1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>w0</td>
<td>k1</td>
<td>k2</td>
<td>H(k1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>w0</td>
<td>k1</td>
<td>k2</td>
<td>H(k1</td>
<td></td>
</tr>
</tbody>
</table>

Generator: the algorithm takes the last bit of the garbling of each input wire for point-and-permute. Later garbling of the output wire is assigned and 3 signals are transmitted to the evaluator due to row reduction.

Evaluator: only one SHA-1 operation will be used due to point-and-permute. The 1-out-of-2 oblivious transfer makes sure that the evaluator will obtain the garbling without knowing the real value.

Implementation Details

For demonstration on FPGAs, we modified the ObliVM* software protocol to output structure of garbled circuits for our implementation and the garbling on every input and output wire.

ObliVM is a programming framework which can compile the target program into the garbled circuits primitive and also perform the computation while generating the circuits.

Each garbling is 80 bits and GateID is 64 bits. SHA-1 algorithm will take 224 bits input and provide the corresponding output after 80 clock cycles, since there are 80 rounds in the standard SHA-1 algorithm. The testbench generated by ObliVM proves the feasibility of our FPGA implementation.

- SHA-1 operation run much faster on an FPGA than high level software code;
- parallelism has the potential of enhancing the performance by a factor of four on the generator side. This shows the promising utilization of FPGA compared with software for larger garbled circuits.

Future Work

- FPGA implementation of larger circuits to demonstrate the feasibility of our approach;
- Code generation automation tools so that a non-expert can explore parallelism using fully customized hardware on FPGAs.