Garbled Circuits for Preserving Privacy in the Datacenter

Xin Fang
Dept of Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
fang.xi@husky.neu.edu

Stratis Ioannidis
Dept of Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
ioannidis@ece.neu.edu

Miriam Leeser
Dept of Electrical and Computer Engineering
Northeastern University
Boston, MA, USA
mel@coe.neu.edu

ABSTRACT

Garbled Circuits (GC) are a flexible way to support Secure Function Evaluation, and thus preserve the privacy of user data. We believe that GC is well suited to acceleration using FPGAs in the datacenter. In this paper we present some preliminary results.

1. INTRODUCTION

Privacy is an increasing concern as more of our transactions become digital and data is shared with others. The field of Secure Function Evaluation (SFE) addresses the problem of evaluating functions without being able to determine the original values of the function's inputs. Popular techniques for secure function evaluation include garbled circuits (GC), originally introduced by Yao [3]. In general, SFE is much more computationally intensive than function evaluation in the clear, and for this reason has not been widely adopted. Recently, implementations of GC have appeared, but these are orders of magnitude slower than simple function evaluation. We believe that SFE can be substantially accelerated by FPGAs in the datacenter, making these techniques usable and more widely adopted.

In this paper we consider two parties holding private data, who wish to know the outcome of a function without revealing the raw data on which the function is evaluated. This can easily be extended to several parties with private data and one evaluator who computes the function. We make the following assumptions: 1) All parties know the function to be evaluated and wish to share the result, but individuals do not wish to reveal their raw data. 2) We assume an *honest but curious* trust model where all parties agree to follow the protocol, but are free to try to infer as much information as possible within these constraints. In addition, we take advantage of *oblivious transfer* which allows parties to share data without revealing that data. The canonical example of garbled circuits is known as the millionaire's problem: two millionaires wish to know which of them is worth more without exchanging their actual worth. This is the type of problem that GC is intended to solve. More relevant problems in today's society might be to determine the average blood pressure of a group of individuals without any of them revealing their medical records.

2. FPGA OVERLAY ARCHITECTURE

GC allows many types of problems to be solved provided they can be represented as a Boolean circuit, which encompasses a very large range of functions. Recent techniques for garbling make the evaluation more efficient, by representing such a circuit as a network or AND gates and XOR gates, where garbling of the XOR gates is considered *free*. Inputs to a circuit are first encrypted, resulting in an 80 bit value. Thus free XOR gates garble 80 bits. AND gates consist of four encryption cores. For our implementation, we use SHA-1 cores working on 80 bit inputs; this encryption is strong enough within the context of GC. We make use of an open source SHA-1 core [1] and use 512 bits of input and 80 cycles to evaluate an AND.

The design of GC is well suited to a coarse grained overlay architecture where a sea of garbled AND gates is implemented on the FPGA along with XORs. Note that a problem such as matrix multiplication may require tens of thousands of garbled AND gates. An overlay architecture allows us to rapidly reuse the garbled gates for different AND gates in the same layer of a GC, different layers, or different applications, without requiring recompilation of the FPGA implementation. On the garbling side, a problem is first translated to a Boolean circuit, then assigned to layers of gates for evaluation. The CPU transfers inputs as well as an assignment of gates to realizations in the overlay. Output of the FPGA consists of encrypted values to represent each AND gate that are transferred to the evaluator.

3. PRELIMINARY RESULTS

Our goal is to run GC on the Novo-G cluster in Florida. Our current implementation targets a Gidel board with a Stratix V FPGA, one of the boards on the Novo-G cluster. For a range of small examples, including millionaire, hamming distance and sorting (among others) we are seeing 2 to 3 orders of magnitude improvement in run time over a sequential implementation using ObliVM [2]. We use ObliVM both to generate the garbled circuit structure and to validate our results. In the future, we plan to extend our implementation to larger examples as well as multiple nodes in the cluster.

4. REFERENCES

[1] J. Strömbergson. SHA1. https://github.com/secworks/sha1.

[2] X. S. Wang, C. Liu, K. Nayak, Y. Huang, and E. Shi. ObliVM: A programming framework for secure computation. *IEEE Symposium on Security and Privacy (S & P)*, 2015.

[3] A. Yao. How to generate and exchange secrets. In *Foundations of Computer Science*, pages 162–167, 1986.