Journal of Network and Computer Applications 108 (2018) 1–19

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Recent advancements in garbled computing: How far have we come towards achieving secure, efficient and reusable garbled circuits

Ahsan Saleem [a], Abid Khan [a], Furqan Shahid [a], M. Masoom Alam [a,*], Muhammad Khurram Khan [b]

[a] Department of Computer Science, COMSATS Institute of Information Technology, Park Road, Chak Shahzad, Islamabad, Pakistan
[b] Center of Excellence in Information Assurance (CoEIA), King Saud University, Saudi Arabia

ARTICLE INFO

Keywords:
Garbled circuits
Secure computation
OT
Reusable GC
Privacy free garbling

ABSTRACT

Secure function evaluation (SFE) is a cryptographic protocol that facilitates participants to compute a function on their private inputs in such a way that privacy of their inputs is preserved. In early 80's Andrew Yao proposed a solution for secure function evaluation known as garbled circuits based on series of Boolean gates and encryption of truth tables. The approach was thought to be theoretically very appealing, however, the scheme was not practical due to sever implementation issues in its efficiency and reusability. However, since then the garbled circuits have evolved and the common notion of impractical garbled circuit have changed greatly mainly due to great deal of interest shown and efforts made by the researchers in a bid to make Yao's garbled circuits practical. In this paper we have analyzed some of the most significant contributions of researchers in enhancing various aspects of garbled circuits and provided a comprehensive comparative analysis and discussed the pros and cons of such schemes. Furthermore, we provide a classification of existing garbled circuit schemes in the form of a comprehensive thematic survey. We conclude our paper by providing new research directions in this domain for new researchers.

1. Introduction

Secure function evaluation is one of the promising concepts in the field of cryptography that is basically aimed at allowing two or more participants to evaluate a function on their private inputs. The protocol commits privacy to each of the participants inputs. Roots of this concept can be traced back to early 80s, when Yao proposed two-party secure function evaluation (Yao, 1982, 1986), for allowing two participants to evaluate a given function on their private inputs. In Yao's construction, the concept of oblivious transfer (OT) (Halpern and Rabin, 1983; Even et al., 1985) is used where parties can evaluate garbled circuits by exchanging encrypted information without learning about input of other parties. The idea was enhanced by Goldreich et al. (1987) from "two-party" to "multi-party" that allowed multiple participants to evaluate a function on their private inputs. After the invention of Multi-Party Computation (MPC), the Two-Party Computation (2PC) can positively be thought as a selected case of MPC. Although, MPC also covers 2PC, 2PC has its own special importance because of its applications in different areas of cryptography, including secure function evaluation, one-time programs, key dependent message security, verifiable computation, and homomorphic/functional encryption (Vinayagamurthy, 2014) but not limited to techniques mentioned above. Because of this reason we can find lots of research work on garbled circuits including, implementations, enhancements, and optimizations targeting special case i.e., 2PC rather MPC. Yao's protocol for secure function evaluation requires translation of the underlying function into its equivalent Boolean Circuit and then garbling each gate of the circuit. Since the core concept is "garbling" that's why it is sometimes referred as "Yao garbling scheme". In this paper, we also have used this term many times in the survey that refers to "Yao two-party secure function evaluation protocol" (see Tables 1–3).

Yao garbling process consists of mainly four steps, i.e., transformation of the function (to be evaluated) into its equivalent Boolean Circuit (consisting of gates AND, OR, XOR etc.). Then inputs and outputs of each gate are garbled by replacing 0 and 1 with random values. On the basis of those garbled values, a garbled table is created for each gate; these tables reveal nothing about their corresponding gates (either it is AND, OR etc.). However, the output of the final gate is not garbled.

* Corresponding author.
E-mail address: masoom.alam@comsats.edu.pk (M. Masoom Alam).

https://doi.org/10.1016/j.jnca.2018.02.006
Received 27 November 2017; Received in revised form 18 January 2018; Accepted 6 February 2018
Available online 12 February 2018

Table 1
Comparative analysis of implementation techniques.

Paper	Security Model	Scope	Specification Language	Optimization Strategies	Major Limitations
(Henecka et al., 2010)	Semi-honest	Compiler, Run-time Environment	TASTYL (Python based language)	Hybrid SFE, Fast Multiplication Tables (Karatsuba), Shifting of most computations to less time critical phase (setup phase), Free XOR, Garbled Row Reduction, Pipelined approach, Efficient OT extension (by Ishai et al.)	Evaluating the same function on different inputs requires compilation each time, Integration with other applications will require changing in the source code (due to absence of APIs)
(Kreuter et al., 2013)	Both Semi-honest and Malicious	Compiler, two Interpreters (one in C, second in Java)	Any language can be used (works on byte code)	Online circuit compression, Lazy gate generation, Ephemeral circuits, Free XOR gates, used priority queue to handle loops rather than stack, Dead gate removal, Optimization is performed before loops are un-rolled	
(Songhori et al., 2015)	Semi-honest	Libraries (HDL Synthesis) for circuit generation	HDL Languages (Verilog, VHDL) or High-Level Languages with HLS compatibility (like C, C++, Python)	Sequential representation of circuits, Exploitation of HDL synthesis libraries to generate circuits with minimum number of non-XOR gates	The whole circuit is created first and then evaluation starts; Memory requirement for storing whole of the circuit (in case of very large functions) may be a challenge
(Huang et al., 2011)	Semi-honest	Java based Library, Execution Environment	JAVA will be used for function and circuit specifications	The whole circuit is not loaded into memory at once; but a pipelined approach is used	Programmer is required to be able to design Boolean circuits
(Kreuter et al., 2012)	Malicious	Compiler, Optimizer	A un-typed language with static scoping	Parallelization of all steps of the protocol, Circuit Pipelining, AES-NI, Removal of redundant, unused, and identity gates	Scalability, Function description language is difficult to use, Memory requirement is linear with the circuit size
(Pinkas et al., 2009)	Semi-honest, Covert, and Malicious	Based on compiler of Fairplay; but used their own runtime environment	Same as Fairplay (SFDL)	Replacing component circuits by simple combination of circuits, Removing the components whose output is either always be 0 or always be 1, Garbled Row Reduction	
(Holzer et al., 2012)	Semi-honest; Malicious	Compiler (CBMC-GC)	ANSI C	Syntactic preprocessing, Exploiting CBMC to leave placeholders for basic operations (like addition and multiplication), Populating these placeholders with components having minimum number of non-XOR gates	
(Malka, 2011)	Semi-honest	JAVA based library (API)	JAVA	Circuits (here called components) are compiled only once, Client and Server hold only a small part of circuit in memory at a time, Circuit is garbled and evaluated in segments rather than as a whole	
(Malkhi et al., 2004)	Limited case of malicious adversaries	Compiler, High-level language, Run time environment	Secure Function Definition Language (SFDL); a subset of Pascal or C	Code simplification, Duplicate code removal, Dead code elimination, Efficient OT protocol (by Naor and Pinkas)	Scalability; Even a circuit with millions of gates require days to complete