Future Generation Computer Systems 62 (2016) 54–65

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Garbled computation in cloud☆

Yongge Wang [a,*], Qutaibah M. Malluhi [b], Khaled MD Khan [c,b]

[a] Department of SIS, UNC Charlotte, NC, USA
[b] KINDI Center for Computing Research, Qatar University, Qatar
[c] Department of Computer Science and Engineering, Qatar University, Qatar

HIGHLIGHTS

- A linear code based garbled circuits scheme is proposed.
- New concept of all-or-nothing privacy for garbled circuits schemes.
- Much more efficient that existing FHE schemes.

ARTICLE INFO

Article history:
Received 26 April 2015
Received in revised form
9 August 2015
Accepted 4 November 2015
Available online 11 December 2015

Keywords:
Cloud computing
Garbled circuits
Reusable garbled computation

ABSTRACT

With the wide adoption of cloud computing paradigm, it is important to develop appropriate techniques to protect client data privacy in the cloud. Encryption is one of the major techniques that could be used to achieve this gaol. However, data encryption at the rest along is insufficient for secure cloud computation environments. Further efficient techniques for carrying out computation over encrypted data are also required. Fully homomorphic encryption (FHE) and garbled circuits are naturally used to process encrypted data without leaking any information about the data. However, existing FHE schemes are inefficient for processing large amount of data in cloud and garbled circuits are one time programs and cannot be reused. Using modern technologies such as FHE, several authors have developed reusable garbled circuit techniques in recent years. But they are not efficient either and could not be deployed at a large scale. By relaxing the privacy definition from perfect forward secrecy to all-or-nothing privacy, we are able to design efficient reusable garbled circuits in this paper. These reusable garbled computation techniques could be used for processing encrypted cloud data efficiently.

1. Introduction

Cloud computing techniques become pervasive and users begin to store their private encrypted data in cloud services. In order to take full advantage of the cloud computing paradigm, it is important to design efficient techniques to protect client data privacy in the cloud. From a first look, encryption at rest seems to be a feasible solution to address these challenges. Lots of companies are trying hard to secure cloud data with encryption techniques. But a truly optimal solution is still far from us since encryption is not a good or even an acceptable solution for cloud data storage. If encryption at rest is the only solution, then the functionality of cloud computing is limited to: encrypt data at the user's location, transmit encrypted data to the cloud, and then bring the data back to the user's location for decryption before being used locally. This is against one of the cloud computing paradigms "moving computation is cheaper than moving data" (see, e.g., [1]). Indeed, in many scenarios, it is less expensive to store data locally than in the cloud. So using the cloud for data-storage without the capability of processing these data remotely may not be an economic approach.

This shows the importance of developing techniques for processing encrypted data at the cloud without downloading them to the local site. A natural solution is to use garbled computing techniques such as garbled circuits or fully homomorphic encryption schemes. Yao [2] introduced the garbled circuit concept which allows computing a function f on an input x without leaking any information about the input x or the circuit used for the computation of $f(x)$. Since then, garbled circuit based protocols have been used

☆ Work was done when the first author was at Qatar University. The work reported in this paper is supported by Qatar Foundation Grants NPRP8-2158-1-423 and NPRP X-063-1-014.

* Corresponding author.
E-mail addresses: yonwang@uncc.edu (Y. Wang), qmalluhi@qu.edu.qa (Q.MD Malluhi), k.khan@qu.edu.qa (K.M. Khan).

http://dx.doi.org/10.1016/j.future.2015.11.004
0167-739X/

in numerous places and it has become one of the fundamental components of secure multi-party computation protocols. Garbled circuit techniques could be further developed as one of the potential techniques allowing computation over encrypted cloud data. However, there are two disadvantages in Yao's approach. Firstly, Yao's garbled circuit is not reusable. Secondly, using a garbled circuit to evaluate an algorithm on encrypted data takes the worst-case runtime of the algorithm on all inputs of the same length since Turing machines are simulated by circuits via unrolling loops to their worst-case runtime, and via considering all branches of a computation.

It has been an open question for 30 years of designing reusable garbled circuits and reusable garbled RAMs or Turing machines. Recently, Goldwasser et al. [3] and Garg et al. [4] (see also Brakerski and Rothblum [5], Barak et al. [6], and Pass et al. [7]) constructed reusable garbled circuits by using techniques of computing on encrypted data such as fully homomorphic encryption (FHE) schemes by Gentry [8], and attribute-based encryption (ABE) schemes for arbitrary circuits by Gorbunov, Vaikuntanathan and Wee [9] and Sahai and Waters [10]. Goldwasser et al. [11] also constructed reusable garbled Turing machines by employing techniques of FHE, witness encryption (WE) schemes by Garg et al [12], and the existence of SNARKs (Succinct Non-interactive Arguments of Knowledge) by Bitansky et al. [13]. However, both of the current reusable garbling schemes for circuits and Turing machines in Goldwasser et al. [11,3] and Garg et al. [4] are inefficient unless there would be a breakthrough in improving the efficiency of FHE implementations dramatically. Under current FHE implementation techniques, it is more efficient to use Yao's one-time garbled circuits than to use the garbling schemes in [4,11,3].

In Goldwasser et al.'s garbling scheme [11,3], the owner of a circuit C encrypts the circuit C using an ideal cipher (e.g., AES) and gives a copy of this encrypted circuit $\bar{C} = \mathtt{E.Enc}(\mathtt{sk}, C)$ to the evaluator. Each time when the circuit owner wants the evaluator to calculate $C(x)$, the circuit owner creates a homomorphic encryption scheme public key $\mathtt{hpk}$ and a corresponding private key $\mathtt{hsk}$. Using the newly created public key $\mathtt{hpk}$, the circuit owner calculates the homomorphic encryption cipher texts c for the input bits x and key bits $\mathtt{sk}$ bit by bit and constructs a Yao's one-time garbled circuit D for decrypting the homomorphic encryption scheme by integrating the private key $\mathtt{hsk}$ within D. The circuit owner gives (D, c) to the evaluator. The evaluator uses a universal circuit to homomorphically decrypt the circuit C from $\mathtt{E.Enc}(\mathtt{sk}, C)$ and then runs C on the input x homomorphically. After the evaluation, the evaluator obtains the homomorphic encryption ciphertext $E_{\mathtt{hpk}}(C(x))$ of $C(x)$. In order for the evaluator to decrypt $E_{\mathtt{hpk}}(C(x))$, the circuit owner uses an attribute based encryption scheme to send corresponding labels for the garbled circuit D so that the evaluator will be able to decrypt $E_{\mathtt{hpk}}(C(x))$ to $C(x)$. It is easy to show that these garbled circuits are reusable since the only published circuit is the universal circuit which contains no information about the protected circuit C which is encrypted using a secure symmetric encryption scheme.

Though [11,3] give an affirmative answer to the reusable garbled circuit open problem, it is still open to design efficient reusable garbled circuits. The schemes in Goldwasser et al. [11,3] are inefficient from two aspects: expensive homomorphic encryption schemes are used for the input encryption and there is a slow process of integrating the universal circuit and the FHE bootstrapping operations into the ABE ciphertexts.

The main performance cost for FHE is due to the fact that all existing FHE schemes are based on "noisy" encryption schemes where homomorphic operations increase the noise in ciphertexts. For example, Goldwasser et al.'s solution [3] is based on the Learning with Errors (LWE) problem. After a homomorphic operation (e.g., a circuit gate evaluation) is performed on the ciphertexts, Gentry's [8] bootstrapping technique is used to refresh the ciphertexts by homomorphically computing the decryption function on encrypted secret key, and bringing the noise of the ciphertexts back to acceptable levels. The bootstrapping operation is the main bottleneck in FHE implementation due to the complexity of homomorphic decryption. Recently, Halevi and Shoup [14] reported a bootstrapping algorithm that takes 30 min to re-encrypt a single-bit ciphertext. More recently, Halevi and Shoup [15] reported a re-encryption procedure under 5.5 min (for a security level of 76 bits). By performing an additional single bit operation before bootstrapping, Ducas and Micciancio [16] reported some fast algorithms for bootstrapping NAND gate though it is not clear whether it can be extended to other circuit gate types.

The above discussion shows the motivation for designing practical reusable garbled circuits without the use of FHE schemes. For Yao's one-time garbled circuits, the security of input and circuit privacy has been studied by several authors such as Lindell and Pinkas [17] and Bellare, Hoang, and Rogaway [18]. However, the privacy requirement for reusable garbled circuits and reusable garbled Turing machines has not been fully understood yet. In Goldwasser et al.'s work [11,3], the simulation based privacy definition requires perfect forward secrecy. That is, the leakage of one input x will not leak any information about other inputs $y \neq x$. This kind of perfect forward secrecy is standard in most cryptographic protocol designs. However, it may be unnecessarily strong for reusable garbled circuits/Turing machine applications. In this paper, we consider all-or-nothing privacy with reusability for reusable garbled circuits and reusable garbled Turing machines. By all-or-nothing privacy with reusability, we mean that if the circuit/Turing machine owner does not disclose any individual input x (in plaintext) on which the garbled circuit/Turing machine has evaluated, then zero information about other inputs (on which the garbled circuit/Turing machine has evaluated) is leaked. However, if the circuit/Turing machine owner discloses any input x on which the garbled circuit/Turing machine has evaluated, then the values of all other inputs (on which the garbled circuit/Turing machine has evaluated) will be leaked. Under this security definition of all-or-nothing privacy with reusability, this paper shows the following results without using FHE schemes:

1. There exists an efficient universal circuit U_C such that for any circuit C, $U_C(\bar{C}, \bar{x})$ outputs $C(x)$ where $\bar{C} = \mathtt{E.Enc}(\mathtt{sk}, C)$, $\bar{x} = \mathtt{E.Enc}(\mathtt{sk}, x)$, and E is an ideal cipher.

We conclude this section with some notations. Unless specified otherwise, we will use $q = 2^m$ and our discussions will be based on the field $GF(q)$ throughout this paper. Bold face low case letters such as $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$, $\mathbf{e}$, $\mathbf{f}$, $\mathbf{g}$ are used to denote row or column vectors over $GF(q)$. It should be clear from the context whether a specific bold face letter represents a row or column vector. For a string $x \in GF(q)^n$, we use $x[i]$ to denote the ith element of x. That is, $x = x[0] \cdots x[n-1]$. We use $x \in_R GF(q)$ to denote that x is randomly chosen from $GF(q)$ with the uniform distribution. We use κ to denote the security parameter, $p(\cdot)$ to denote a function p that takes one input, and $p(\cdot, \cdot)$ to denote a function p that takes two inputs. A function f is said to be negligible in an input parameter κ if for all $d > 0$, there exists K such that for all $\kappa > K, f(\kappa) < \kappa^{-d}$. For convenience, we write $f(\kappa) = \mathrm{negl}(\kappa)$. Two ensembles, $X = \{X_\kappa\}_{\kappa \in N}$ and $Y = \{Y_\kappa\}_{\kappa \in N}$, are said to be computationally indistinguishable if for all probabilistic polynomial-time algorithm D, we have

$$|Pr[D(X_\kappa, 1^\kappa) = 1] - Pr[D(Y_\kappa, 1^\kappa) = 1]| = \mathrm{negl}(\kappa).$$

Throughout the paper, we use probabilistic experiments and denote their outputs using random variables. For example, $\mathrm{Exp}^{\mathrm{real}}_{E,A}(1^\kappa)$ represents the output of the real experiment for scheme E with adversary A on security parameter κ.