

Oblivious Infinite Derivable Function Evaluation and Its Application

Pang Shuting¹, Tang Chunming^{1,2}, Hao Zhifeng^{2,3}

¹ School of Mathematics and Information Science, Guangzhou University, Guangzhou, China
Email: christine0201@163.com

² State Key Laboratory of Information Security, Institute of Software Science, Beijing, China

³ Faculty of Computer, Guangdong University of Technology, Guangzhou, China

Abstract—Based on oblivious polynomial evaluation for floating-point numbers and Taylor series, we construct an oblivious function for any infinite derivable function, which generalizes oblivious function evaluation protocol. Then we simplify oblivious neural computing from our oblivious function evaluation.

Index Terms—Oblivious evaluation, Taylor series, Neural computing

I. INTRODUCTION

Oblivious function evaluation protocol, which was first suggested by Moni Naor and Benny Pinkas[4], is an efficient instrument of solving the problem of oblivious function evaluation. Assume that there are two parties Alice and Bob. Alice has a polynomial P , while Bob who has a value x , wants to cooperate with Alice to get $P(x)$ such that Alice learns nothing about x and Bob learns only what can be inferred from $P(x)$. Many remarkable results of OPE (Oblivious Polynomial Evaluation) protocol have been published, such as based on intractability assumptions that are closely related to noisy polynomial reconstruction[4], and based on oblivious transfer only[9]. Not only over finite fields, but over floating-point number[9], OPE protocol is practical in real-world applications.

Data mining is a recently emerging crossed field. Confidentiality issues are concerned. Assume that two parties have private databases respectively, and they wish to cooperate by computing some data mining algorithm on the union of their databases. Similarly to the problem of OPE, in the privacy preserving data mining, neither party is willing to divulge any of the contents to the other. After R. Agrawal, R. Srikant [5] and Y. Lindell, B. Pinkas [10] proposed the notion of privacy preserving data mining respectively, security constraints were added to several machine learning techniques: decision trees[10], neural networks[9], support vector machines[7], naïve bayesian classifiers[3], belief networks[11], clustering[6], and so on.

For the confidentiality issues of the neural network, Y.C.Chang and C.J.Lu [9] has proposed oblivious neural computing and oblivious neural learning which are based

on OPE protocol and OCE(Oblivious Circuit Evaluation) protocol, where one party has a neural network and the other, with some training sets, wants to train the neural network in an oblivious way. Then C.Orlandi, A.Piva and M.Barni proposed their oblivious neural network computing via homomorphic encryption[2], which is based on Paillier cryptosystem.

In neural network, different activation functions cause the nerve cell has different information processing properties, which is one of three main elements that determine the whole property of neural network, therefore the research of activation function is very important and fundamental.

For the oblivious evaluation of the activation function, Y.C.Chang and C.J.Lu [9] only discussed the hyperbolic tangent function

$$f(z) = a \tanh(bz) = a \left(1 - \frac{2}{1 + e^{2bz}} \right).$$

But it's not unique. As the general model of neural computing, activation function should be adopted according to the actual situation.

In this paper, we construct oblivious function evaluation protocol for any infinite derivable function, and use it to realize oblivious neural computing. The OPE protocol for floating-point numbers, private scalar product protocol and Taylor series are our main tools.

II. PRELIMINARIES

Let r denote a small number such that power series can be convergent in $(-r, r)$. Fix a security parameter τ , so that numbers about $2^{-\tau}$ are considered negligible.

A. Oblivious Transfer

A 1-out-of-2 oblivious transfer(OT_1^2) protocol involves two parties, the sender and the receiver. The sender's input is a pair (x_0, x_1) and the receiver's input is a bit $\sigma \in \{0,1\}$. The protocol is such that the receiver learns x_σ (and nothing else) and the sender learns nothing.

Similarly one can define OT_1^n for any $n \geq 3$, or OT_k^n for any $k \geq 2$ and $k \ll n$. It is known that OT_1^2 is proved to be correct and secure, and the existence of OT_1^2 implies the existence of OT_1^n or OT_k^n .

This work were supported by Foundation of National Natural Science(China) (10871222)

B. Oblivious Polynomial Evaluation

A protocol for oblivious polynomial evaluation (OPE) involves a sender and a receiver. The sender's input is a polynomial P of degree n over some finite field \mathbf{F} and the receiver's input is an element $\hat{x} \in \mathbf{F}$ (the degree n of P is public). The protocol is such that the receiver obtains $P(\hat{x})$ without learning anything else about the polynomial P , and the sender learns nothing.

C. Floating-Point Number System

A floating-point number is a rational number

$$b = \pm \sum_{j=1}^{2m} b_j 2^{m-j}$$

for some m , with $b_j \in \{0,1\}$. Let $\langle m \rangle$ denote the floating-point number system containing all such numbers together with standard arithmetic operations.

D. An OPE protocol for floating-point numbers

Let $k = (n+1)m + \tau + 1$ and $l = k + \log(2nm)$. Parties agree on the floating-point system $\langle k \rangle$ for random numbers, and the floating-point system $\langle l \rangle$ for all arithmetics. Let $v_{ij} = 2^{m-j} \hat{x}^i$.

Y.C.Chang and C.J.Lu [9] has proved the security of the following protocol I.

E. Private Scalar Product Protocol

A protocol for private scalar product (PSP) involves two partners, Alice and Bob. They have their vectors $X = (x_1, \dots, x_n)$ and $Y = (y_1, \dots, y_n)$ respectively, and want to compute the scalar product $X \cdot Y$ privately, that is, nobody can know the output.

W.Du and M.J.Atallah has proposed two protocols for PSP problem in Ref. [8]. Combining with our problem, we just need their first protocol as our main tool which is shown in the Protocol II.

PROTOCOL I OBLIVIOUS POLYNOMIAL EVALUATION PROTOCOL

1. Bob prepares $2nm$ 3-tuple $(r_{ij}, v_{ij} + r_{ij}, -v_{ij} + r_{ij})$, $i = 1, \dots, n$, $j = 1, \dots, 2m$, with each r_{ij} chosen randomly from $\langle k \rangle$.
2. For each 3-tuple $(r_{ij}, v_{ij} + r_{ij}, -v_{ij} + r_{ij})$, Alice runs an OT_1^3 with Bob to get r_{ij} if $a_{ij} = 0$, $v_{ij} + r_{ij}$ if $a_{ij} = 1 \wedge a_i > 0$, and $-v_{ij} + r_{ij}$ otherwise.
3. Alice sends to Bob the sum of a_0 and those $2nm$ values she got. Bob subtracts $\sum_{i,j} r_{ij}$ from it to obtain $P(\hat{x})$.

PROTOCOL II PRIVATE SCALAR PRODUCT PROTOCOL

- Inputs: Alice has a vector $X = (x_1, \dots, x_n)$, and Bob has a vector $Y = (y_1, \dots, y_n)$.
- Outputs: Alice gets $X \cdot Y + v$ where v is a random scalar known to Bob only.
1. Alice and Bob agree on two numbers p and m , such that p^m is large enough.
 2. Alice generates m random vectors V_1, \dots, V_m , such that

$$X = \sum_{j=1}^m V_j.$$

3. Bob generates m random numbers r_1, \dots, r_m , such that

$$v = \sum_{j=1}^m r_j.$$

4. For each $j = 1, \dots, m$, Alice and Bob conduct the following sub-steps:

- (a) Alice generates a secret random number k , $1 \leq k \leq p$.
- (b) Alice sends (H_1, \dots, H_p) to Bob, where $H_k = V_j$, and the rest of H_i 's are random vectors. Because k is a secret number known only to Alice, Bob does not know the position of V_j .

- (c) Bob computes $Z_{j,i} = H_i \cdot Y + r_j$ for $i = 1, \dots, p$.

- (d) Using the 1-out-of- p Oblivious Transfer protocol, Alice gets $Z_j = Z_{j,k} = V_j \cdot Y + r_j$, while Bob learns nothing about k .

5. Alice computes $u = \sum_{j=1}^m Z_j = X \cdot Y + v$.

III. OBLIVIOUS INFINITE DERIVABLE FUNCTION EVALUATION

A. Taylor Series of Infinite Derivable Function

Assume that f is the function having arbitrary degree derivative at the point x_0 , for every x satisfying $|x - x_0| < r$ such that

$$\lim_{n \rightarrow \infty} R_n(x) = 0,$$

then f can be expanded into Taylor series in x_0 's neighborhood, namely,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + \dots$$

Specially, the expansion of f at $x_0 = 0$ can be written as

$$f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \dots + \frac{f^{(n)}(0)}{n!}x^n + \dots,$$

which is called Maclaurin series.

B. Oblivious Infinite Derivable Function Problem

Here we want to carry out infinite derivable function evaluation in an oblivious way between Alice and Bob. Suppose now Alice has an infinite derivable function $f(\cdot)$, and Bob has \hat{x} . After our oblivious protocol, Bob can get the value $f(\hat{x})$ without learning anything else about Alice's function, while Alice also learns nothing about \hat{x} .

Since every infinite derivable function can be expanded into Taylor series, especially Maclaurin series at $x_0 = 0$, we can change the problem of oblivious infinite derivable function evaluation into oblivious polynomial evaluation, that is, it's easy to solve the oblivious infinite derivable function evaluation problem, just with Alice's function replaced by a polynomial. In fact, lots of functions are computed by using Taylor series when we use computer to compute it.

C. Oblivious Infinite Derivable Function Evaluation

Here we discuss the situation of two power series expansion. Assume Alice holds an infinite derivable

function f , and Bob holds $\hat{x} \in \langle m \rangle$.

1. Maclaurin Series with Order n

We take n as a positive integer that is not large, then f can approximate to a sum function, namely

$$f(x) \approx f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \dots + \frac{f^{(n)}(0)}{n!}x^n.$$

Let

$$a_0 = f(0), \quad a_i = \frac{f^{(i)}(0)}{i!} \quad (i = 1, \dots, n) \in \langle m \rangle,$$

$f(x)$ can be written as an approximate polynomial

$$P(x) = \sum_{i=0}^n a_i x^i.$$

For each i , let

$$|a_i| = \sum_{j=1}^{2m} a_{ij} 2^{m-j}, \quad \text{with } a_{ij} \in \{0,1\}.$$

We will use the oblivious polynomial evaluation protocol for floating-point numbers, which contains OT_1^3 that can be implemented by 2 executions of OT_1^2 . Note that, if a_i ($i = 0, \dots, n$) is an irrational number, we just need to choose an approximate rational number to replace it for our floating-point number system. It is shown in the above Protocol III.

2. Taylor Series with Order n

In general, when $x_0 \neq 0$, f can be expanded into

PROTOCOL III
OBLIVIOUS FUNCTION EVALUATION PROTOCOL FOR MACLAURIN
EXPANSION

1. Alice expands her function f at the point $x_0 = 0$ into Maclaurin series with order n . Let

$$a_0 = f(0), \quad a_i = \frac{f^{(i)}(0)}{i!} \quad (i = 1, \dots, n) \in \langle m \rangle,$$

then $P(x) = \sum_{i=0}^n a_i x^i \approx f(x)$.

2. Bob runs an OPE protocol for floating-point numbers with Alice to get the approximation value of $f(\hat{x})$.

Taylor series at $x = x_0$, where $x \in (x_0 - r, x_0 + r)$. Similarly to the situation of Maclaurin series, we take the first n items as the approximate value of $f(x)$, and simplify it as follow,

$$\begin{aligned} f(x) &\approx f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + \dots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n \\ &= [f(x_0) + \frac{f'(x_0)}{1!}C_1(-x_0) + \dots + \frac{f^{(n)}(x_0)}{n!}C_n(-x_0)^n] \\ &\quad + [\frac{f'(x_0)}{1!}C_1^0 + \frac{f''(x_0)}{2!}C_2^1(-x_0) + \dots + \frac{f^{(n)}(x_0)}{n!}C_n^1(-x_0)^{n-1}]x \\ &\quad + [\frac{f''(x_0)}{2!}C_2^0 + \dots + \frac{f^{(n)}(x_0)}{n!}C_n^{n-2}(-x_0)^{n-2}]x^2 + \dots + [\frac{f^{(n)}(x_0)}{n!}]x^n \\ &= \left[\sum_{j=0}^n \frac{(-1)^j f^{(j)}(x_0)}{j!} x_0^j \right] + \left[\sum_{j=1}^n \frac{(-1)^{j-1} f^{(j)}(x_0)}{!(j-1)!} x_0^{j-1} \right] x + \dots + \\ &\quad \left[\sum_{j=i}^n \frac{(-1)^{j-i} f^{(j)}(x_0)}{i!(j-i)!} x_0^{j-i} \right] x^i + \dots + \left[\sum_{j=n}^n \frac{(-1)^{j-n} f^{(j)}(x_0)}{n!(j-n)!} x_0^{j-n} \right] x^n \\ &= \sum_{i=0}^n \left[\sum_{j=0}^n \frac{(-1)^{j-i} f^{(j)}(x_0)}{i!(j-i)!} x_0^{j-i} \right] x^i. \end{aligned}$$

Let $a_i = \sum_{j=0}^n \frac{(-1)^{j-i} f^{(j)}(x_0)}{i!(j-i)!} x_0^{j-i}$, $i = 0, 1, \dots, n$, then

Alice's function $f(x)$ can be approximated to a polynomial $P(x) = \sum_{i=0}^n a_i x^i$. Oblivious function evaluation protocol for Taylor expansion is shown in the Protocol IV.

IV. OBLIVIOUS NEURAL COMPUTING

A. BP-Neural Network Computing

The application of BP-neural network is the most common, which is 2-layer feedforward neural networks with back-propagation learning. It contains three layers, which are input layer, internal layer and output layer, where the internal layer has T nodes, with the t 'th node having a weight vector $V_t = (v_{1t}, v_{2t}, \dots, v_{st})$, and the output layer has L nodes, with the l 'th node having a weight vector $W_l = (w_{1l}, w_{2l}, \dots, w_{tl})$.

Each node is associated with an activation function

PROTOCOL IV
OBLIVIOUS FUNCTION EVALUATION PROTOCOL FOR TAYLOR
EXPANSION

1. Bob chooses randomly a value x_0 from the interval of $(\hat{x} - r, \hat{x} + r)$, and sends it to Alice.

2. Alice expands her function f at the point $x = x_0 \neq 0$ into Taylor series with order n . Let

$$a_i = \sum_{j=0}^n \frac{(-1)^{j-i} f^{(j)}(x_0)}{i!(j-i)!} x_0^{j-i} \in \langle m \rangle, \quad i = 0, 1, \dots, n,$$

then $P(x) = \sum_{i=0}^n a_i x^i \approx f(x)$.

3. Bob runs an OPE protocol for floating-point numbers with Alice to get the approximation value of $f(\hat{x})$.

$f(x)$ (Being practical in general, we use Sigmoid function which is non-linear and continuous, as the activation function.). The network takes an input vector $X = (x_1, x_2, \dots, x_s)$ and produces an output vector $O = (o_1, o_2, \dots, o_L)$ in the following way, otherwise an expected output vector $d = (d_1, d_2, \dots, d_L)$ exists in some network models.

For the internal layer,

$$y_t = f(V_t \cdot X), \quad t = 1, \dots, T.$$

For the output layer,

$$o_l = f(W_l \cdot Y), \quad l = 1, \dots, L, \quad Y = (y_1, y_2, \dots, y_T).$$

B. Oblivious Neural Computing

Here we show that how to carry out neural computing in an oblivious way between Alice and Bob. Suppose that Alice has a neural network which is trained to some degree and she uses it to serve the classification requests from Bob, while Bob with a set of training examples wants to help Alice's neural network get better or maybe for his classification result. They don't want the other to know anything about their own secret neither.

After running the protocol V, Alice and Bob obtain r_{t0} , r_{t1} respectively, such that $r_{t0} + r_{t1} \approx y_t$. Moreover, Alice and Bob obtain $R_0 = (r_{10}, r_{20}, \dots, r_{T0})$,

$R_1 = (r_{11}, r_{21}, \dots, r_{T1})$ respectively, such that $R_0 + R_1 = Y$ after running T times of the protocol V .

Our protocol V is based on scalar product protocol and OPE protocol, which are proved to be secure. We will run T times of Protocol V to get the output of T nerve cells for the internal layer, which contains one execution of Scalar Product protocol and one execution of OPE protocol for floating-point numbers.

As for the output layer, we can just modify the first step because the vector Y does not belong to Bob yet, instead of sharing by Alice and Bob. So,

Protocol V
OBLIVIOUS NEURAL COMPUTING FOR THE INTERNAL LAYER

1. Alice's input is $V_t = (v_{1t}, \dots, v_{St})$, Bob's input is $X = (x_1, x_2, \dots, x_S)$. They run PSP protocol that outputs random shares c_{t0} and c_{t1} such that

$$c_{t0} + c_{t1} = V_t \cdot X.$$
2. Bob randomly chooses a value $\hat{c}_{t1} \in (c_{t1} - r, c_{t1} + r)$, and sends it to Alice.
3. Alice computes $x_0 = c_{t0} + \hat{c}_{t1}$, and expand her function f into Taylor series $P(x) = \sum_{i=0}^n a_i x^i$ at the point x_0 , where

$$a_i = \sum_{j=0}^n \frac{(-1)^{j-i} f^{(j)}(x_0)}{i!(j-i)!} x_0^{j-i} \in \langle m \rangle, i = 0, 1, \dots, n.$$
4. Alice randomly chooses r_{t0} , and defines the following polynomial

$$Q(z) = P(c_{t0} + z) - r_{t0}.$$
5. Alice and Bob execute an Oblivious Polynomial Evaluation protocol for Floating-Point Numbers with Alice inputting Q and Bob inputting c_{t1} , in which Bob obtains $r_{t1} = Q(c_{t1})$.

$$W_t' \cdot Y = W_t' \cdot (R_0 + R_1) = W_t' \cdot R_0 + W_t' \cdot R_1,$$

replacing by Alice inputting $W_t = (w_{1t}, \dots, w_{Tt})$ and Bob inputting $R_1 = (r_{11}, \dots, r_{T1})$, Alice only needs to compute $W_t' \cdot R_0$ locally and adds it into her random share.

V. SUMMARY AND FUTURE WORK

In this paper, we have studied oblivious two-party function evaluation protocol for any infinite derivable function which is based on Taylor series. What's more, we apply it into oblivious neural computing as the activation function.

Apart from applying our oblivious function evaluation into neural computing of BP network, there are many other models in neural network, such as SOM, ART,

LVQ, and so on. A future direction is how to combine secure two-party computation with these neural network models, then form many more secure neural computing and secure neural learning.

REFERENCES

- [1] A. C. Yao, "How to generate and exchange secrets", Proc. 27th Ann. IEEE Symp. Foundations of Computer Science, 1986, pp. 162–167.
- [2] C. Orlandi, A. Piva, M. Barni, "Oblivious neural network computing via homomorphic encryption", EURASIP Journal on Information Security, vol. 2007, pp. 1-11.
- [3] M. Kantarcioglu, J. Vaidya, "Privacy preserving naive bayes classifier for horizontally partitioned data". IEEE Workshop on Privacy Preserving Data Mining, 2003.
- [4] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation", Proc. 31st Ann. ACM Symp. Theory of Computing, 1999, pp. 245–254.
- [5] R. Agrawal, R. Srikant, "Privacy-Preserving Data Mining", Proceedings of SIGMOD Conference on Management of Data, May 2000, pp. 439-450.
- [6] S. Jha, L. Kruger, P. McDaniel, "Privacy Preserving Clustering", 10th ESORICS, 2005.
- [7] S. Laur, H. Lipmaa, T. Mielikainen, "Cryptographically private support vector machines", KDD'06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge discovery and data mining, 2006, pp. 618–624.
- [8] W. Du and M. J. Atallah, "Privacy-preserving cooperative statistical analysis", Proceeding of the 17th Annual Computer Security Applications Conference, New Orleans, Louisiana, USA, December 10-14 2001.
- [9] Y. C. Chang and C. J. Lu, "Oblivious polynomial evaluation and oblivious neural learning", Theoretical Computer Science 341(1), 2005, pp. 39–54.
- [10] Y. Lindell, B. Pinkas, "Privacy preserving data mining", Advances in Cryptology - CRYPTO'00, volume 1880 of Lecture Notes in Computer Science 1880, 2000, pp. 36–54.
- [11] Z. Yang, R. N. Wright, "Improved privacy-preserving bayesian network parameter learning on vertically partitioned data", ICDEW'05: Proceedings of the 21st International Conference on Data Engineering Workshops, 2005