Verifiable Delegation of Polynomials

Jun Ye[1], Haiyan Zhang[1], and Changyou Fu[2]
(Corresponding author: Jun Ye)
School of Science, Sichuan University of Science & Engineering[1]
School of Computer Science, Sichuan University of Science & Engineering[2]
No. 180, School Street, Huixing Road, Sichuan 643000, P.R. China
(Email: yejun@suse.edu.cn)
(Received Jan. 10, 2015; revised and accepted July 4, 2015)

Abstract

Verifiable computation allows a computationally weak client to outsource evaluation of a function on many inputs to a powerful but untrusted server. The client invests a large amount of off-line computation in an amortized manner to obtain an encoding of its function which is then given to the server. The server returns both the evaluation result of the function on the client's input and a proof with which the client can verify the correctness of the evaluation using substantially less effort than doing the evaluation on its own from scratch. In this paper a verifiable delegation of polynomials is proposed based on the integer factorization problem. In the computation procedure, the computation polynomial and the verification polynomial are distinguishable, the wrong result and the result of other inputs will incur a validation failure . And last, the client can verify the result efficiently.

Keywords: Cloud computing, integer factorization, verifiable computation, verifiable delegation

1 Introduction

Cloud computing [8, 13, 18, 19, 21] is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud computing has provided plenty of convenience for the resource-constrained clients. Outsourced computations are widely used due to the rise of cloud computing. The complex computing tasks of users whose computing resources are limited can be outsourced to the cloud server. For instance, a large number of real-time updated data should be computed, but the resource-constrained clients are unable to deal with. The computations have to be outsourced to cloud server, nevertheless, in this situation, clients lose the ability to control their data which may be sensitive and highly interest related. The growing desire to outsource computational tasks from a relatively weak client to a computationally more powerful servers and the problem of dishonest workers who modify their clients' software to return plausible results without performing the actual computation motivated the formalization of the notion of Verifiable Computation [6, 15, 16].

Verifiable computation enables a computer to offload the computation of some function, to other probably untrusted servers, while maintaining verifiable results. The servers evaluate the function and return the result with a proof that the computation of the function was carried out correctly.

To ensure that the computation results are correct, the server must provide the results together with a certificate of its correctness. In the progress of outsouring computation, cryptographic techniques [20] are often used. For the resource-constrained clients, the verification of such correctness proof must be much easier than the original computation. If the verification takes more time of computation, the client could perform the computation on its own without interacting with the server. So verifiable computation should at least satisfy the flowing three basic requirements.

1) Server cannot cheat the client with a random value without computing the outsourced function.

2) Server cannot cheat the client with the computing result of other input values.

3) The verification of client should be efficient.

The main results of this paper are two aspects, the function obfuscation technic and the secure scheme for verifiable delegation of polynomials. The function obfuscation technique which is based on the large integer factorization will make an efficient computation polynomial mix in the outsourcing polynomials, and due to the difficulty of large integer factorization the server cannot recognize the polynomial which is mixed in. The secure scheme for verifiable delegation of polynomials is based on this technic, with the mixed efficient computation polynomial client can easily verify the result returned by server.

1.1 Related Work

In 2001, secure outsourcing for scientific computing and numerical calculation are studied for the first time by Atallah, Pantazopoulos and Rice [2], and they put forward a lot of suitable camouflage technology for scientific computing, such as, matrix multiplication, inequality, linear equations, etc. These technologies ensure the privacy of user's data, but this does not solve the problem of the verifiability of computing results. In 2005, the formal security definition of outsourcing has been presented for the first time by Hohenberger and Lysyanskaya [12] , and two provably secure outsourcing schemes are proposed, the basic outsourcing scheme of modular exponentiation and the CCA2 security outsourcing encryption scheme. In 2008, Benjamin and Atallah [5] constructed a verifiable secure outsourcing scheme for linear algebraic calculation by using semantic security based homomorphic encryption. In 2009, Gentry [10] proposed the fully homomorphic encryption scheme for the first time based on ideal lattices. But the efficiency is low.

In 2010, Atallah and Frikken [1] proposed a single server verifiable outsourcing scheme based on Shamir secret sharing scheme, and Gennaro, Gentry and Parno proposed an outsourcing computation scheme for arbitrary function F with non-interactive verification based on fully homomorphic encryption. In 2011, Chung, Kalai and Liu [7] proposed an outsourcing model, and puts forward the idea of memory delegation. In this scheme the user can change the outsourcing data in memory, however the data flow cannot be arbitrarily long. And Benabbas, Gennaro and Vahlis [4] studied the problem of computing on large datasets that are stored on an untrusted server, the weak client can make retrieval and update queries. This is the first construction that relies on a "constant-size" assumption, and does not require expensive generation of primes per operation. In 2012, Parno, Raykova and Vaikuntanathan [16] proposed the verifiable multi-function computation scheme. However, the user can distribute the data to the server only once, and the relevant information should be stored locally. And in the same year, Seitkulov [17] put forward a new verifiable camouflage computation scheme, which can be used to achieve verifiable secure outsourcing for abstract equations, Cauchy problem with secret parameters, boundary value problems with secret boundary conditions and some nonlinear equations. And Fiore and Gennaro [9] presented new protocols for publicly verifiable secure outsourcing of evaluation of high degree polynomials and matrix multiplication based on the closed form efficient pseudorandom functions. In 2013, Backes, Fiore and Reischuk [3] proposed novel cryptographic techniques that solve the above problem for the class of computations of quadratic polynomials over a large number of variables. Papamanthou and Shi and Tamassia [14] also have studied public verification and considered the case of polynomial evaluation.

Features comparisons between our scheme and some recent schemes are list in Table 1.

Table 1: Comparisons with related works

	Full Security	Constant Assumption
Scheme [14]	$\surd$	$\times$
Schemes [16] + [11]	$\times$	$\surd$
Our Scheme	$\surd$	$\surd$

1.2 Organization of this Paper

The organization of this paper is as follows. Some preliminaries are given in Section 2. The algorithms of verifiable computation are given in Section 3. Then in Section 4 we give our protocol of verifiable delegation of polynomials. The security analysis is given in Section 5. The efficiency of our scheme analysis is given in Section 6. Finally, conclusion will be made in Section 7.

2 Preliminaries

Some definitions and technics are listed in this section, which will be used in the following sections.

2.1 Negligible Function

A negligible function is a function $negl(x)$ such that for every positive integer c there exists an integer N_c such that for all $x > N_c$ such that

$$\mid negl(x) \mid < \frac{1}{x^c}$$

Equivalently, we have the following definition. A function $negl(x)$ is negligible, if for every positive polynomial $poly(\cdot)$ there exists an integer $N_{poly} > 0$ such that for all $x > N_{poly}$

$$\mid negl(x) \mid < \frac{1}{poly(x)}$$

2.2 Integer Factorization Problem

Given a number $n = pq$, where p and q are two large prime numbers, it is difficult to factorize n.

This problem has many different versions. Here we introduce a decisional problem.

Given $n = pq$, and an x with the corresponding y, it is difficult to determine that y satisfies which of the following equations:

$$y = ax \bmod p \quad \text{or} \quad y = ax \bmod p^*$$

where, $p^* < n$ is a random prime number.

It can be described as the indistinguishable way, see the following experiment.

Let IFP be the integer factorization problem, $\mathcal{A}$ is a PPT adversary, $d \in \{0, 1\}$. **Gen** denotes the generation

algorithm, and **Compute** denotes the computation algorithm.

Experiment $\mathbf{Exp}^{p,p^*}_{IFP,\mathcal{A}}[n,x,y]$
$(p,p^*,n=pq,a,x) \leftarrow \mathbf{Gen}(1^k)$ and
$b_0=p, b_1=p^*$
For $i=1$ to t
$x_i \leftarrow \mathcal{A}(n,x_1,y_1,\cdots,x_{i-1},y_{i-1})$
$y_i = ax_i \bmod b_d \leftarrow \mathbf{Compute}(x_i)$
$x^* \leftarrow \mathcal{A}(n,x_1,y_1,\cdots,x_t,y_t)$
$y^* \leftarrow \mathbf{Compute}(x^*)$
$d' \leftarrow \mathcal{A}(n,x_1,y_1,\cdots,x_t,y_t,x^*,y^*)$
If $d'=d$, output 1.
Else output 0.

The advantage of an adversary $\mathcal{A}$ in the above experiment is defined as:

$$Adv^{ind}_{IFP,\mathcal{A}}(n,p,p^*) = |\Pr[d'=d]-\tfrac{1}{2}|$$

The factorization of integer $n=pq$ is a difficult problem means

$$Adv^{ind}_{IFP,\mathcal{A}}(n,p,p^*) \le negl(k)$$

where $negl(\cdot)$ is a negligible function.

2.3 Homomorphic Encryption

Homomorphic encryption is a form of encryption which allows some computations (such as addition, multiplication and exponentiation) to be carried out on ciphertext and obtain an encrypted result the decryption of which matches the result of operations performed on the plaintexts.

Assume $E(\cdot)$ is an encryption algorithm, and $D(\cdot)$ is an decryption algorithm, fully homomorphic encryption should satisfy the following properties.

$(\sigma_{x_1},\sigma_{x_2}) \leftarrow E(x_1,x_2)$, then $D(\sigma_{x_1}+\sigma_{x_2}) = x_1+x_2$.

$(\sigma_{x_1},\sigma_{x_2}) \leftarrow E(x_1,x_2)$, then $D(\sigma_{x_1}\cdot\sigma_{x_2}) = x_1\cdot x_2$.

Fully homomorphic encryption is useful in outsourcing computations.

Given $\sigma_x \leftarrow E(x)$, $\sigma_y = f(\sigma_x)$, then $y=D(\sigma_y)$, which satisfies $y=f(x)$.

3 Verifiable Computation

A verifiable computation scheme is with two parties client and server. Client outsources the computation of a function f to an untrusted server. Client expects server to evaluate the function on an input and server returns a result with a proof that the result is correct. Then the client verifies that the result provided by the server is indeed correct about the function on the input. In this scenario, client should verify the result efficiently with much less cost of computation resources.

A verifiable computation scheme is defined by the following algorithms:

KeyGen$(f,k) \rightarrow (PK,SK)$**:** Based on the security parameter k, the key generation algorithm generates a key pair (PK,SK) for the function f. PK is provided to the server, and client keeps SK.

ProGen$_{SK}(x) \rightarrow (\sigma_x, V_x)$**:** The problem generation algorithm is run by client to uses SK to encode the input x as σ_x which is given to server, and a verification key V_x which is kept private by client.

Compute$_{PK}(\sigma_x) \rightarrow (\sigma_y)$**:** Given PK and σ_x, the algorithm is run by the server to compute an encoded version of the output σ_y.

Verify$_{SK}(V_x,\sigma_y) \rightarrow (y \cup \perp)$**:** Using the secret key SK, the verification key V_k, and the encoded output σ_y, the algorithm returns the value $y=f(x)$ or $\perp$ indicating that σ_y does not equal to $f(x)$.

A verifiable computation scheme should be correct, secure and efficient. A verifiable computation scheme $\mathcal{VC}$ is correct if the algorithms allow the honest server to output values that will pass the verification.

Definition 1 (Correctness)**.** *A verifiable computation scheme is correct if the algorithms allow the honest server to output values that will pass the verification. That is, for any x, f and any $(PK,SK) \leftarrow KeyGen(f,k)$, if $(\sigma_x,V_x) \leftarrow ProGen_{SK}(x)$, $(\sigma_y) \leftarrow Compute_{PK}(\sigma_x)$, then $f(x) \leftarrow Verify_{SK}(V_x,\sigma_y)$ holds with all but negligible probability.*

In other words, for an input x and a given function f, a malicious server should not be able to convince the verification algorithm on output σ'_y such that $\sigma'_y \neq f(x)$. We use the following experiment to describe this.

Experiment $\mathbf{Exp}^{Vf}_{\mathcal{A}}[\mathcal{VC},f,k]$
$(PK,SK) \leftarrow \mathbf{KeyGen}(f,k)$
For $i=1$ to q
$x_i \leftarrow A(PK,x_1,\sigma_{x,1},V_{x,1},\cdots,x_{i-1},\sigma_{x,i-1}V_{x,i-1})$
$(\sigma_{x,i},V_{x,i}) \leftarrow \mathbf{ProGen}_{SK}(x_i)$
$x^* \leftarrow \mathcal{A}(PK,\sigma_{x,1},V_{x,1},\cdots,\sigma_{x,q},V_{x,q})$
$(\sigma_{x^*},V_{x^*}) \leftarrow \mathbf{ProGen}_{SK}(x^*)$
$\sigma'_y \leftarrow \mathcal{A}(PK,\sigma_{x,1},V_{x,1},\cdots,\sigma_{x,q},V_{x,q},\sigma_{x^*},V_{x^*})$
$y' \leftarrow (PK,V_{x^*},\sigma'_y)$
If $y' \neq \perp$ and $y' \neq f(x^*)$, output 1.
Else output 0.

A verifiable computation scheme is secure if an incorrect output cannot be accepted. That is, the probability that the verification algorithm accepts the wrong output value for a given input value is negligible.

For a verifiable computation scheme, we define the advantage of an adversary $\mathcal{A}$ in the above experiment as:

$$Adv^{Vf}_{\mathcal{A}}(V,f,k) = \Pr[\mathbf{EXP}^{Vf}_{\mathcal{A}}[V,f,k]=1]$$

Then we get the definition of security.

Definition 2 (Security)**.** *A verifiable computation scheme is secure if for any function f, and any PPT adversary $\mathcal{A}$, that*

$$Adv_{\mathcal{A}}^{Vf}(V, f, k) \leq negl(k)$$

where $negl(\cdot)$ is a negligible function.

In the verifiable computation scheme the time for verifying the output must be much smaller than the time to compute the function.

Definition 3 (Efficiency)**.** *A verifiable computation scheme is efficient, if the time required for* ***Verify***(V_x, σ_y) *is $o(T)$, where T is the time required to compute $f(x)$.*

4 Verifiable Delegation of Polynomials

In this section, we give the polynomial obfuscation technique based on integer factorization problem, and give our implementation scheme.

4.1 Obfuscation Technic

The polynomial

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_dx^d, a_i \in Z_p, 0 \leq i \leq d$$

which is a high degree polynomial, is outsourced to server. Client asks the server to compute the function on the value of x. In this scenario, the client must be able to verify the correctness of the result efficiently.

For the secure and efficient verification, an efficient computing polynomial $v(x)$ is chosen, and a verifiable polynomial $F(x) = af(x) + v(x)$ is constructed. This should satisfy the following requirements.

1) Server cannot get any information about a from $f(x)$ and $F(x)$.
2) $v(x)$ cannot be identified by server.
3) $f(x)$ and $F(x)$ are indistinguishable.

We use the following technic, which is based on integer factorization problem, to achieve the requirements.

Client selects a prime q randomly (which satisfies $|p| = |q|$), and computes $n = pq$. Then client randomly chooses $a \in Z_p^*$. $F(x)$ is generated as

$$\begin{aligned} F(x) &= af(x) + v(x) \bmod n \\ &= b_0 + b_1x + b_2x^2 + \cdots + b_dx^d \bmod n \end{aligned}$$

where,$b_i \in Z_p, 0 \leq i \leq d$.

Proposition 1. *Server cannot get any information about a from $f(x)$ and $F(x)$.*

Proof. For $a = (b_i)^{-1}a_i \bmod p(i = 0, 1, \cdots, d)$, if p is known, then a can be uniquely determined. Server just knows $n = pq$, where q is unknown. If p is undetermined, there are p choices of a, so a is secure. □

Proposition 2. *The efficient computation polynomial $v(x)$ is secure in the computation process.*

Proof. Consider the following experiment.

Experiment $\mathbf{Exp}_{IND,\mathcal{A}}^{p,p^*}[n, x, y]$
$(p, p^*, n = pq, a, x) \leftarrow \mathbf{Gen}(1^k)$ and
$b_0 = p, b_1 = p^*$
For $i = 1$ to t
$x_i \leftarrow \mathcal{A}(n, x_1, y_1^{(1)} = ax_1 \bmod p,$
$y_1^{(2)} = ax_1 \bmod p^*, \cdots, x_{i-1},$
$y_{i-1}^{(1)} = ax_{i-1} \bmod p,$
$y_{i-1}^{(2)} = ax_{i-1} \bmod p^*)$
$y_i^{(1)} = ax_i \bmod p, y_i^{(2)} = ax_i \bmod p^*$
$\leftarrow \mathbf{Compute}(x_i)$
$x^* \leftarrow \mathcal{A}(n, x_1, y_1^{(1)}, y_1^{(2)} \cdots, x_t, y_t^{(1)}, y_t^{(2)})$
$y^* = ax^* \bmod b_d \leftarrow \mathbf{Compute}(x^*)$
$d' \leftarrow \mathcal{A}(n, x_1, y_1^{(1)}, y_1^{(2)} \cdots, x_t, y_t^{(1)}, y_t^{(2)}, x^*, y^*)$
If $d' = d$, output 1.
Else output 0.

The advantage of an adversary $\mathcal{A}$ in the above experiment is defined as:

$$Adv_{IND,\mathcal{A}}^{ind}(n, p, p^*) = |\Pr[\mathcal{A}(p) = 1] - \Pr[\mathcal{A}(p^*) = 1]|$$

Let

$$v(x) = r_0 + r_1x + r_2x^2 + \dot{+}r_dx^d, r_i \in Z_p, 0 \leq i \leq d$$

then

$$b_i = aa_i + r_i \bmod p.$$

We also have the equation

$$b_i = aa_i \bmod p^*$$

where $p^* \neq p$. For there exists a prime p^* such that $p^*|(aa_i - b_i)$, so $aa_i - b_i = kp^*$, then the equation $b_i = aa_i \bmod p^*$ is existent.

Due to the difficulty of large integer factorization, we know

$$Adv_{IFP,\mathcal{A}}^{ind}(n, p, p^*) \leq negl(k)$$

so

$$\begin{aligned} & Adv_{IND,\mathcal{A}}^{ind}(n, p, p^*) \\ = \; & |\, Pr[\mathcal{A}(p) = 1] - Pr[\mathcal{A}(p^*) = 1] \,| \leq negl(k). \end{aligned}$$

For n cannot be factorized, p is unknown, the server cannot distinguish $b_i = aa_i + r_i \bmod p$ from $b_i = aa_i \bmod p^*$. So the coefficient r_i is secure.

Thus $v(x)$ cannot be identified by server.

□

Proposition 3. *$f(x)$ and $F(x)$ are indistinguishable.*

Proof. $f(x)$ and $F(x)$ are two polynomials with same degree d, and $b_i = aa_i + r_i \bmod p, 0 \leq i \leq d$. There exist $b, r_i' \in Z_p$, such that $a_i = bb_i + r_i' \bmod p$.

For p is unknown and $a, r_i, 0 \leq i \leq d$ are safe, a_i and b_i are indistinguishable. Thus $f(x)$ and $F(x)$ are indistinguishable. □

4.2 Verifiable Scheme with Efficient Computing Functions

We assume x is an encoded input and y is an encoded output, the function f is a homomorphic encrypted function.

Client wants to computes $y = f(x) \bmod p$ ($f(x)$ is a high degree polynomial), he/she delegates it to a server. And the client can verify the correctness of the result.

Initialization. Client selects a prime q randomly, and computes $n = pq$. Then client randomly chooses $e(0 < e < d)$ and $r_1, r_2 \in Z_p^*$. Client keeps p, q, e, r_1, r_2 and send n to server.

Delegation. Client sends two polynomials

$$y = f(x) \bmod n$$

and

$$y = af(x) + x^e + r_1 + r_2 x \bmod n$$

to server.

Verification. Server sends $y_1 = f(x) \bmod n$ and $y_2 = af(x)+x^e+r_1+r_2x \bmod n$ to client. Client computes $m = x^e + r_1 + r_2x \bmod n$ and verifies whether the following equation holds.

$$y_2 = ay_1 + m \bmod n.$$

If this equation does not hold, that means the server gives the wrong answer, y_1 is not correct. If the equation holds, that means y_1 is the right answer, and client can get the final result by computing

$$y = y_1 \bmod p.$$

5 Security Analysis

5.1 Security of Parameters

Theorem 1. *The probability that server can get p from the computation process is no more than $\frac{1}{s}$, where s satisfies $s \ln s = n$.*

Proof. Server gets two polynomials

$$y_1 = f(x) \bmod n$$

and

$$y_2 = af(x) + x^e + r_1 + r_2x \bmod n$$

so server knows the coefficients $a_i (i = 0, 1, \cdots, d)$ of $y_1 = a_0+a_1x+\cdots+a_dx^d$, and the coefficients $b_i (i = 0, 1, \cdots, d)$ of $y_2 = b_0 + b_1x + \cdots + b_dx^d$. The random number a is unknown.

Comparing the coefficients, server can get $b_i = aa_i \bmod p$, and server can compute $a = (b_i)^{-1}a_i \bmod p$. But a is unknown, for every prim p there exists one a such that $a = (b_i)^{-1}a_i \bmod p$. Server should find a number a, such that

$$a = (b_i)^{-1}a_i \bmod p \quad (i = 0, 1, \cdots, d).$$

There are about s prime numbers which are less than n, where s satisfies $s \ln s = n$. So the probability server can get p is no more than $\frac{1}{s}$. If p is a 512 bit prime. the probability is negligible.

On the other hand, server knows $n = pq$, so server can get p form decomposing n. But by the difficulty of integer factorization, the probability that p can be obtained from the factorization of n is negligible.

In these two aspects, the probability that server can get p from the computation process is no more than $\frac{1}{s}$. □

Theorem 2. *The probability that server can obtain a from the computation process is at most* $\max(\frac{1}{p}, \frac{1}{s})$.

Proof. For $a = (b_i)^{-1}a_i \bmod p (i = 0, 1, \cdots, d)$, if p is known, then a can be uniquely determined. The probability that p can be uniquely determined is no more than $\frac{1}{s}$, where s satisfies $s \ln s = n$, so the probability a can be determined is also $\frac{1}{s}$.

In another way server just knows $n = pq$, where q is unknown. If p is undetermined, there are p choices of a, so the probability that a can be determined is at most $\frac{1}{p}$.

So the probability that server can obtain a from the computation process is at most $\max(\frac{1}{p}, \frac{1}{s})$. □

Theorem 3. *The mixed polynomial $x^e + r_1 + r_2x$ is safe in the computation process.*

Proof. $x^e + r_1 + r_2x$ is mixed in the polynomial $y_2 = af(x) + x^e + r_1 + r_2x \bmod n$, and server do not know a, e, r_1, r_2 and p. Server can determine the function parameters through some special values. The equations are as follows.

$$\begin{aligned}
y_2^{(0)} &= af(0) + 0 + r_1 + 0 \\
y_2^{(1)} &= af(1) + 1 + r_1 + r_2 \\
y_2^{(2)} &= af(2) + 2^e + r_1 + 2r_2 \\
y_2^{(3)} &= af(3) + 3^e + r_1 + 3r_2 \\
\cdots &\quad \cdots \\
y_2^{(m)} &= af(m) + m^e + r_1 + mr_2
\end{aligned}$$

where, $a, 2^e, 3^e, \cdots, m^e, r_1, r_2$ are unknown parameters. There are $m + 1$ equations with $m + 2$ unknown parameters, so server cannot gain the parameters from these equations.

If server guesses a value of the parameters, the other parameters can be computed out. Even if server gets $2^e, 3^e, \cdots, m^e$, the unknown variable e is still safe for the difficulty of discrete logarithm problem.

Server also guess out e, r_1, r_2 from the p values in Z_p, the probability is $\frac{1}{p^3}$.

So the mixed polynomial $x^e + r_1 + r_2x$ is safe in the computation process. □

Theorem 4. *The two polynomials $y_1 = f(x) \bmod n$ and $y_2 = af(x) + x^e + r_1 + r_2x \bmod n$ are indistinguishable.*

Proof. The probability p can be determined is no more than $\frac{1}{s}$, and a is at most $\max(\frac{1}{p}, \frac{1}{s})$. $x^e+r_1+r_2x$ is mixed in the polynomial $y_2 = af(x) + x^e + r_1 + r_2x \bmod n$. For the mixed polynomial $x^e + r_1 + r_2x$ is safe in the computation process, the two polynomials $y_1 = f(x) \bmod n$ and $y_2 = af(x)+x^e+r_1+r_2x \bmod n$ are indistinguishable. □

This theorem means that server cannot distinguish which polynomial is the one client wants to compute and which one is for verification.

5.2 No Fraudulence

Theorem 5. *The probability that the random values which the server provides without the evaluation of the two polynomials* $y_1 = f(x) \bmod n$ *and* $y_2 = af(x) + x^e + r_1 + r_2x \bmod n$ *can pass the verification is* $\max(\frac{1}{p^4}, \frac{1}{sp^3})$.

Proof. The probability that server can obtain a from the computation process is at most $\max(\frac{1}{p}, \frac{1}{s})$, that is

$$\Pr\{\mathcal{A}(a) = 1\} = \max(\tfrac{1}{p}, \tfrac{1}{s}).$$

The probability that $x^e + r_1 + r_2x$ can be obtained from the computation process is $\frac{1}{p^3}$, that is

$$\Pr\{\mathcal{A}(x^e + r_1 + r_2x) = 1\} = \tfrac{1}{p^3}$$

The two random values of the two polynomials $y_1 = f(x) \bmod n$ and $y_2 = af(x) + x^e + r_1 + r_2x \bmod n$ should satisfy the equation

$$y_2 = ay_1 + m \bmod n$$

so that they can pass the verification. However server does not know a and m, so

$$\Pr\{\mathcal{V}(y_1, y_2) = 1\} = \max(\tfrac{1}{p}, \tfrac{1}{s})\tfrac{1}{p^3} = \max(\tfrac{1}{p^4}, \tfrac{1}{sp^3}).$$

Thus probability that the random values can pass the verification is $\max(\frac{1}{p^4}, \frac{1}{sp^3})$. □

Theorem 6. *The advantage that server uses the result of other input to cheat client is at most* $\frac{e}{p}$.

Proof. Considering the following experiment.

Experiment $\mathbf{Exp}^{x_0}_{PE,\mathcal{A}}(k)$

$X = (x_1, x_2, \cdots x_t) \leftarrow \mathbf{Gen}(1^k)$

$Y = ((y_1^{(1)} = f(x_1),$
$y_2^{(1)} = af(x_1) + x_1^e + r_1 + r_2x_1),$
$(y_1^{(2)} = f(x_2),$
$y_2^{(2)} = af(x_2) + x_2^e + r_1 + r_2x_2), \cdots,$
$(y_1^{(t)} = f(x_t),$
$y_2^{(t)} = af(x_t) + x_t^e + r_1 + r_2x_t))$
$\leftarrow \mathcal{A}(X, (f(x), af(x) + x^e + r_1 + r_2x))$

$x^* \neq x_0 \leftarrow \mathcal{A}(X, Y)$

Return 1, if $af(x^*) + (x^*)^e + r_1 + r_2x^*$
$= af(x^*) + (x_0)^e + r_1 + r_2x_0.$

Else, return 0.

$$Adv_{\mathcal{A},x_0} = \Pr[\mathcal{A}^{x^*}(k) = 1 \mid x^* \leftarrow \mathbf{Gen}(1^k), x^* \neq x_0] = \Pr[Return1]$$

Client wants to compute $y_1 = f(x_1)$, but server gives the values $y_2 = f(x_2)$ and $y_2' = f(x_2) + x_2^e + r_1 + r_2x_2$ $(x_1 \neq x_2)$.

If the equation

$$y_2' = ay_2 + x_1^e + r_1 + r_2x_1$$

holds, the value y_2 can pass verification. That means

$$x_1^e + r_1 + r_2x_1 = x_2^e + r_1 + r_2x_2 \bmod p(x_1 \neq x_2)$$

should hold.

There are at most e values in Z_p can satisfy this equation

$$x_1^e + r_1 + r_2x_1 = m \bmod p.$$

So the probability the equation $x_1^e+r_1+r_2x_1 = x_2^e+r_1+r_2x_2 \bmod p$ $(x_1 \neq x_2)$ holds is at most $\frac{e}{p}$.

Hence,

$$Adv_{\mathcal{A},x_0} = \tfrac{e}{p}$$

□

6 Efficiency Analysis

To compute $f(x) = a_0 + a_1x + a_2x + \cdots + a_dx^d$ client needs $2d$ times multiplications. If the polynomial $f(x)$ is outsourced to server, the client need to compute $x^e+r_1+r_2x$ and verifies wether $y_2 = y_1+x^e+r_1+r_2x$ holds. If the verification passed, client should compute $y = y_1 \bmod p$. In this way, client needs $e+1$ times multiplications and a modular exponentiation computation. If e is chosen much less than d, the computation cost is much smaller than the cost of computing $f(x)$.

The time cost comparisons of verification and computation are as Figure 1. We implement our mechanism using MATLAB language with a version of R2013a. The process is conducted on a computer with Intel(R) Core(TM)i7-3770 CPU processor running at 3.40 GHz, 16 GB RAM.

7 Conclusion

Cloud computing has made a reality of computation outsourcing. A new protocol for publicly verifiable outsourcing of evaluation of high degree polynomials is given in this paper. And we introduce a function obfuscation technic, the secret polynomial can be mixed into the outsourced polynomial by using this technic. This technic can also be used in the information hiding. In the verification phase, the result returned by server can be easily verified by client. And the time cost is much less than the time computing the original polynomial. Client can choose the value of e, such that the computation of verification value can be controlled within the user's computational capabilities.

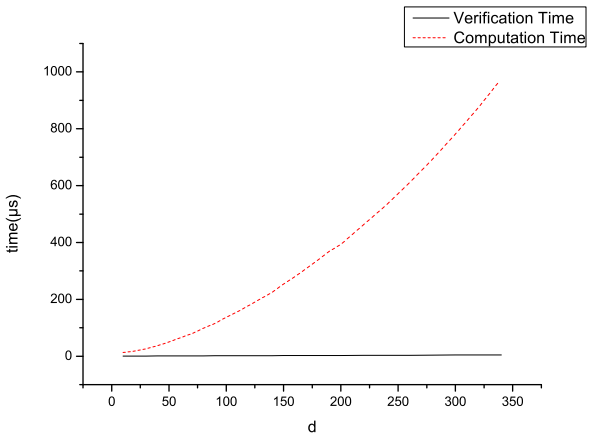

Figure 1: Time cost comparisons of verification and computation

Acknowledgment

This study was supported by the Opening Project of Sichuan Province University Key Laboratory of Bridge Non-destruction Detecting and Engineering Computing; The Science Founding of Artificial Intelligence Key Laboratory of Sichuan Province (No.2014RYJ06); The enterprise information and control technology about internet of things in Sichuan province key laboratory of colleges and universities (No.2014WYJ03); The Scientific Research Fund Project of Sichuan University of Science & Engineering (No.2013KY02). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10)*, pp. 48–59, Springer, Apr. 2010.

[2] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford, "Secure outsourcing of scientific computations," *Advances in Computers*, vol. 54, pp. 215–272, 2001.

[3] M. Backes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 863–874, Springer, Mar. 2013.

[4] S. Benabbas, R. Gennaroand, Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Advances in Cryptology (CRYPTO'11)*, LNCS 6841, pp. 111–131, Springer, 2011.

[5] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Sixth Annual Conference on Privacy, Security and Trust (PST'08)*, pp. 240–245, Springer, Oct. 2008.

[6] S. G. Choi, J. Katz, R. Kumaresan, and C. Cid, "Multi-client non-interactive verifiable computation," in *Proceedings of the 10th Theory of Cryptography Conference (TCC'13)*, LNCS 7785, pp. 499–518, Springer, Mar. 2013.

[7] K. M. Chung, Y. T. Kalaiand, and F. H. Liu, "Memory delegation," in *Proceedings of the 31st Annual Cryptology Conference*, LNCS 6841, pp. 151–168, Springer, Aug. 2011.

[8] S. El-Sayed, H. Kader, M. Hadhoud, and D. AbdElminaam, "Mobile cloud computing framework for elastic partitioned/modularized applications mobility," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 53–63, 2014.

[9] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 501–512, Springer, Oct. 2012.

[10] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Symposium on Theory of Computing (STOC'09)*, pp. 169–178, 2009.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, LNCS 3494, pp. 89–98, Springer, Oct. 2006.

[12] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proceedings of the Second Theory of Cryptography Conference (TCC'05)*, LNCS 3378, pp. 264–282, Springer, Feb. 2005.

[13] X. Ma, J. Li, and F. Zhang, "Refereed computation delegation of private sequence comparison in cloud computing," *International Journal of Network Security*, vol. 17, no. 6, pp. 743–753, 2015.

[14] C. Papamanthou, E. Shi, and R. Tamassia, "Signatures of correct computation," in *Theory of Cryptography*, LNCS 7785, pp. 222–242, Springer, 2013.

[15] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, pp. 238–252, Mar. 2013.

[16] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proceedings of the 9th Theory of Cryptography Conference (TCC'12)*, LNCS 7194, pp. 422–439, Springer, Mar. 2012.

[17] Y. N. Seitkulov, "Verifiable delegation of computation on outsourced data," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 469–482, 2013.

[18] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[19] J. F. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Q. Ma, and X. F. Chen, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Computer Science and Information Systems*, vol. 10, no. 2, pp. 667–684, 2013.

[20] N. Wu and M. Hwang, "Data hiding: current status and key issues," *International Journal of Network Security*, vol. 4, no. 1, pp. 1–9, 2007.

[21] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

Jun Ye received his M.S. degree in Cryptography at the Guilin University of Electronic Technology. He is a Lecturer at the School of Science, Sichuan University of Science & Engineering. He is a doctoral candidate at the Xidian University. His research interests include cryptography and information security. Email: yejun@suse.edu.cn.

Haiyan Zhang received a BS degree in applied mathematics from Sichuan Normal University, China, in 2002, and a MS degree in pattern recognition and intelligent system from Sichuan University of Science & Engineering, China, in 2009. She is currently an instructor with Sichuan University of Science & Engineering. Her research interests include applied mathematics and intelligent system. Email: zhang_petrel@qq.com.

Changyou Fu received a BS degree in communication engineering from University of Electronic Science and Technology of China, in 2000. He is currently a senior experimentalist with Sichuan University of Science & Engineering. His research interests include Internet of Thing and embedded systems. Email: fcybill@163.com.