International Journal of Digital Crime and Forensics
Volume 11 • Issue 4 • October-December 2019

Privacy-Preserving and Publicly Verifiable Protocol for Outsourcing Polynomials Evaluation to a Malicious Cloud

Dawei Xie, Shandong University, Jinan, China
Haining Yang, School of Mathematics, Shandong University, Jinan, China
Jing Qin, Shandong University, Jinan, China
Jixin Ma, Greenwich University, London, UK

ABSTRACT

As cloud computing provides affordable and scalable computational resources, delegating heavy computing tasks to the cloud service providers is appealing to individuals and companies. Among different types of specific computations, the polynomial evaluation is an important one due to its wide usage in engineering and scientific fields. Cloud service providers may not be trusted, thus, the validity and the privacy of such computation should be guaranteed. In this article, the authors present a protocol for publicly verifiable delegations of high degree polynomials. Compared with the existing solutions, it ensures the privacy of outsourced functions and actual results. And the protocol satisfies the property of blind verifiability such that the results can be publicly verified without learning the value. The protocol also improves in efficiency.

KEYWORDS

Blind Verification, Cloud Computing, Large Polynomials, Privacy, Publicly Verifiable

1. INTRODUCTION

It is increasingly common for mobile devices with relatively weak computing power to be used as general computing devices, such as smart phones and netbooks. This trend, coupled with their increasing desire to execute computationally intensive tasks, makes outsourcing computation to the cloud service providers a promising solution. The outsourcing computation enables resource-constrained clients to enjoy almost unlimited computational resources. The clients deliver their computational tasks to cloud service providers and receive computational result from the providers in a pay-per-use manner. Hence, the clients are no longer restricted to their limited CPU, storage and bandwidth. Moreover, outsourcing computation provides economic benefits to the clients and allows them to avoid or minimize up-front IT infrastructure costs.

Despite the tremendous advantages, outsourcing computation raises several new security challenges, which make the clients reluctant to outsource their computations to cloud service providers. The first concern is the correctness of the computational results done by the providers, since the cloud service providers may not be trusted. Their misbehaviors are motivated by financial incentives (e.g., saving the computing resources for other transactions) or caused by hacking and bug in system. It means that there is no guarantee on the correctness of the computational results.

DOI: 10.4018/IJDCF.2019100102
This article, originally published under IGI Global's copyright on October 1, 2019 will proceed with publication as an Open Access article starting on February 2, 2021 in the gold Open Access journal, International Journal of Digital Crime and Forensics (converted to gold Open Access January 1, 2021), and will be distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

In order to address this problem, the notion of Verifiable Computation (VC) was introduced, which enables the clients to verify correctness of the results. Clearly, the verification of correctness must be substantially easier than the computation that was initially outsourced. Next, Parno, Raykova, and Vaikuntanathan (2012) proposed a new notion of Publicly Verifiable Computation (PVC) by extending the definition of VC, which makes the computational results can be verified by any third party besides the delegators. This is important in the contexts where the results have to be checked by several clients who cannot share a secret key in advance. Another concern is the privacy, since the relevant data may contain sensitive information, such as personal health assessment, stock prediction, financial performance analysis and so on, the computational results and the outsourced functions should be hided. Encryption does not fundamentally solve this problem, because it is very difficult or inefficient to perform meaningful operations on ciphertext. One common approach is that the client does some carefully designed local disguise process of the function before sending it to cloud service providers. Consider the privacy of results, blind verifiability is an important property of outsourcing computation such that the results can be publicly verified without learning the value. For example, a financial company delegates a certain computational task to cloud service providers, and it wants to ensure the privacy of the results. Further, efficiency is also a crucial challenge. Thus, an outsourcing computation protocol should satisfy the following four design goals: public verifiability, privacy of result/function, blind verifiability and efficiency.

In this paper, the authors study the particular problem of polynomial evaluation. Among all types of computations, the polynomial function evaluation is an important one due to its wide usage in engineering and scientific problems. For instance, the medical center executes polynomial functions over the personal health data, which uploaded from various wearable devices. Based on these data, the medical center assesses personal health and makes suggestions for people to keep healthy. Also, the securities company utilizes the financial software to analyze the economic performance of stocks, by executing polynomial functions over the past data. Benabbas, Gennaro, and Vahlis (2011) proposed the first practical verifiable computation protocol for high degree polynomial functions, by using the algebraic pseudorandom functions (PRFs). Nevertheless, their protocol does not satisfy public verifiability. In the same line of work, Fiore and Gennaro (2012) devised new algebraic PRFs to develop a publicly verifiable computation protocol for the delegated polynomials. After that, researchers proposed numerous protocols for secure outsourcing of polynomials (e.g., Catalano & Fiore, 2013; Elkhiyaoui, Azraoui, & Molva, 2016; Luo, Yang, & Cong, 2018; Papamanthou, Shi, & Tamassia, 2013; Ye, Zhang, & Fu, 2016; Zhang & Safavi-Naini, 2014).

From the practical perspective, however, the existing protocols for outsourcing polynomials computation still have some limitations, which make it not practical in real world application scenarios. The first limitation is that the actual polynomials and the actual results are revealed to the cloud servers. In many practical delegation scenarios, the privacy of the polynomials and the results must be ensured. The second limitation is that almost all protocols do not satisfy the blind verifiability. Consider a financial company that wishes to delegate the computation of a certain expensive data analysis polynomial function to a cloud server. Since the company will make an investment strategy for the next year based on the results, it would like to ensure the privacy of the polynomial and the results. Meanwhile, the blind verifiability is also a necessary requirement. Once the relevant data are leaked, rivals will speculate on its investment plans and sabotage the investments. The damage could result in huge economic losses for the company. The gaps in the existing protocols motivate the authors to investigate the outsourcing polynomials computation.

In this paper, the authors propose an efficient publicly verifiable outsourcing computation protocol for polynomial functions. The key characteristics of the protocol are that it can ensure the privacy (of the polynomials and the actual results) and support blind verifiability.

1.1. Our Contributions

The main contributions in this paper can be summarized as follows.

- The authors propose a privacy preserving and publicly verifiable protocol for outsourcing polynomials evaluation to a malicious cloud. It can ensure the privacy of outsourced polynomials and actual results, and it supports the blind verifiability. The authors leverage Euclidian division as a basis for our protocol.
- The disguise process, the blind verification and the retrieval for actual results can be executed efficiently.

1.2. Related Works

The subject of Verifiable Computation has already attracted a lot of attention. To begin with, some related works focus on Interactive Proof (Babai, 1985; Goldwasser, Micali, & Racko, 1989) and efficient arguments based on the probabilistically checkable proofs (Kilian,1995; Kilian, 1998). Gennaro, Gentry, and Parno (2010) proposed a non-interactive verifiable computation protocol. In this protocol, a client outsources data and a function evaluation to a cloud server. The client executes a one-time encoding of the function and stores it with the server. This enables the server to not only evaluate the function, but also provide a proof of the correctness. However, the protocol requires the verifier know a secret key in advance, it does not satisfy public verifiability. In order to address this shortcoming, Parno, Raykova, and Vaikuntanathan (2012) extended the definition of Verifiable Computation to introduce a new notion of Publicly Verifiable Computation with the support of Attribute-Based Encryption (ABE). Their protocols explore a connection between Attribute-Based Encryption and the problem of verifiable computation, which enables the results can be verified by any third party. However, the class of functions for which we know an efficient ABE protocol is unfortunately very limited, and their protocol does not work for any arbitrary polynomial time computation. Next, numerous papers focus on verifiable computation of specific functions (e.g., Blanton, Zhang, & Frikken, 2013; Hu & Tang, 2015; Lei, Liao, Member, Huang, & li, 2015; Zhou & li, 2017). Recently, many researchers have presented applications of publicly verifiable computation (Backes, Fiore, & Reischuk, 2013; Song, Wang, & Wang, 2017; Sun, Zhu, & Qin, 2018; Ye, Zhang, & Fu; Zhang, Wei, & Li, 2017).

The verifiable polynomial evaluation also has a large body of prior works. Benabbas, Gennaro, and Vahlis (2011) presented the first practical verifiable computation protocol for high degree polynomial functions, which is based on the algebraic PRFs. In their scheme, a client stores polynomials in the clear with a cloud server as a vector of the coefficients. Catalano and Fiore (2013) proposed a practical protocol for the verifiable polynomial computation with the bounded degree in virtue of homomorphic MACs. In this paper, their first construction relies on the sole assumption that one way functions exist and allows for arbitrary composition (i.e., outputs of previously authenticated computations can be used as inputs for new ones), but it has the drawback that the size of the produced tags grows with the degree of the circuit. Their second construction, relying on the Diffle-Hellman Inversion assumption, offers some features as it allows for very short tags but poses some restrictions on the composition side. Since the use of symmetric encryption algorithm, their protocol does not satisfy the public verifiability. Ye, Zhang, and Fu (2016) was based on the integer factorization problem to propose verifiable computation protocol for polynomial. Above protocols do not satisfy public verifiability, privacy of result/function, blind verifiability as the protocol presented in this paper.

Fiore and Gennaro (2012) devised new algebraic PRFs, also used by Zhang and Safavi-Naini (2014), to develop publicly verifiable protocols. Their protocols enable the computational results can be verified by any third party. Papamanthou, Shi, and Tamassia (2013) proposed a protocol for publicly verifiable polynomials computation, which is based on *signatures for correct computation* (SCC). In this paper, a client outsources a polynomial F to an untrusted cloud server. The server generates a succinct signature σ for the correctness of the computation results. Elkhiyaoui, Azraoui, and Molva (2016) proposed a publicly verifiable protocol of polynomials that improves in efficiency with the support of Euclidean division. Compared with these protocols, the protocol of this paper offers the additional privacy and blind verifiability. (See Table 1)

1.3. Paper Organization

In section preliminaries, the authors recall the computational assumptions and the definitions needed by the paper. In next section, the authors present a protocol for public verifiable polynomial evaluation and some analysis about it. Finally, they conclude their works.

2. PRELIMINARIES

In what follows the authors will denote with $\lambda \in \mathbb{N}$ a security parameter. If S is a set, they denote with $x \xleftarrow{R} S$ the process of selecting x uniformly at random in S. The authors say that a function $negl(\cdot)$ is negligible if it vanishes faster than the inverse of any polynomial. Let $\mathcal{A}$ be a probabilistic polynomial time algorithm. They denote with Dom(F) the domain of F and Ran(F) the range of F.

2.1. Computational Assumptions

2.1.1. Definition 1 (Bilinear Pairing)

Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic groups of prime order p. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with the following properties:

1. *Bilinearity. For any α, $\beta \in \mathbb{Z}_p$, $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$,* there exists an equation $e\left(g^{\alpha}, h^{\beta}\right) = e\left(g, h\right)^{\alpha\beta}$;
2. *Computability. For any $\left(g, h\right) \in \mathbb{G}_1 \times \mathbb{G}_2$,* there exists an efficient algorithm to compute $e\left(g, h\right)$;
3. *Non-degeneracy. If g* is a generator of $\mathbb{G}_1$ and h is a generator of $\mathbb{G}_2$, then $e\left(g, h\right)$ is a generator of $\mathbb{G}_T$.

2.1.2. Definition 2 (n-SDH Assumption)

Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic groups of prime order p, g is a generator of $\mathbb{G}_1$, h is a generator of $\mathbb{G}_2$ and e is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

Then n-Strong Diffie-Hellman assumption (n-SDH) holds, if given the tuple $\left(g, g^{\alpha}, h, h^{\alpha}, \ldots, h^{\alpha^n}\right) \in \mathbb{G}_1^2 \times \mathbb{G}_2^{n+1}$ for $\alpha \xleftarrow{R} \mathbb{F}_p^*$, the probability of producing a pair $\left(\beta, h^{1/(\beta+\alpha)}\right) \in \mathbb{F}_p \setminus \left\{-\alpha\right\} \times \mathbb{G}_2$ is negligible.

Table 1. This Is A features comparison of the protocol for polynomials evaluation with related protocols

Protocols	Public Verifiability	Public Delegation	Result/Function Privacy	Blind Verifiability
Gennaro et al, 2011	×	×	×	×
Fiore et al,2012	√	×	×	×
Papamanthou et al, 2013	√	√	×	×
Zhang et al, 2014	×	×	×	×
Elkhiyaoui et al, 2016	√	√	×	×
Song et al, 2017	√	√	×	×
Our Work	√	√	√	√

2.2. Publicly Verifiable Computation

A publicly verifiable computation protocol is a tuple of given algorithms, the clients delegate the computation of a function F to a cloud server S by these algorithms, in such way that the clients can verify the correctness of the results returned by the server S in public way. Clearly, it is crucial that the work of verification must be substantially smaller than computing the function locally. Otherwise, the clients could perform the computation on itself.

Parno, Raykova, and Vaikuntanathan (2012) defined the publicly verifiable computation (PVC): any verifier (possibly different from the delegator) can verify the correctness of the server's results.

Let $\mathcal{F}$ be a family of functions. The modified PVC protocol between two polynomial time parties: a client C and a server S is defined by the following algorithms:

- $\mathrm{KeyGen}\left(F,1^{\lambda}\right) \rightarrow \left(PP, PK_F, RK_F, EK_F\right)$: On input a function $F \in \mathcal{F}$ and a security parameter λ, the randomized key generation algorithm produces public parameters PP, a public key PK_F, a retrieval key RK_F that will be used to retrieve the actual results and an evaluation key EK_F which will be handed to the server S.
- $\mathrm{ProbGen}\left(PK_F, PP, x\right) \rightarrow \left(\sigma_x, VK_x\right)$: Given a value $x \in \mathrm{Dom}(F)$, the public key PK_F and the public parameters PP, the delegator runs this algorithm to produce an encoding σ_x of x and a public verification key VK_x that will be used to check the results.
- $\mathrm{Compute}\left(EK_F, \sigma_x\right) \rightarrow \left(\sigma_y, \pi\right)$: Given the evaluation key EK_F together with the encoding σ_x, this algorithm is run by the server S to compute an encoding σ_y of $F\left(x\right)$ and a proof π of its correctness.
- $\mathrm{BVerify}\left(VK_x, PP, \sigma_y, \pi\right) \rightarrow T_{\sigma_y}$: On input the public verification key VK_x, the public parameter PP, the encoded output σ_y and the proof π, this blind verification algorithm is run by any party to return a token T_{σ_y}. If the output σ_y is valid, then this algorithm returns $T_{\sigma_y} = \left(accept, S\right)$; otherwise, it returns $T_{\sigma_y} = \left(reject, S\right)$.
- $\mathrm{Retrieve}\left(RK_F, T_{\sigma_y}, \sigma_y\right) \rightarrow y \cup \perp$: Run by the clients holding RK_F to retrieve the actual result $y = F\left(x\right)$ or an error $\perp$.

2.2.1. Definition 3 (Correctness)

A publicly verifiable computation protocol for a family of functions $\mathcal{F}$ is correct, if for any function $F \in \mathcal{F}$ and any $x \in$ Dom(F):

If $\mathrm{ProbGen}\left(PK_F, PP, x\right) \rightarrow \left(\sigma_x, VK_x\right)$ *and* $\mathrm{Compute}\left(EK_F, \sigma_x\right) \rightarrow \left(\sigma_y, \pi\right)$, *then* $\Pr\left(\mathrm{BVerify}\left(VK_x, PP, \sigma_y, \pi\right) \rightarrow \left(accept, S\right)\right) = 1$.

Informally, a publicly verifiable computation protocol is correct, if the values generated by the problem generation algorithm allows the honest server S to output results that will be verified correctly.

For any publicly verifiable computation protocol, the authors define following experiment:

Algorithm 1. Soundness experiment of publicly verifiable protocol: $\left(PP, PK_F, EK_F\right) \leftarrow \mathcal{O}_{\mathrm{KeyGen}}\left(F,1^{\lambda}\right)$;

$\mathcal{A} \rightarrow x$;

$$\left(\sigma_x, VK_x\right) \leftarrow \mathcal{O}_{\text{ProbGen}}\left(PK_F, PP, x\right);$$

$$\mathcal{A} \rightarrow \left(\sigma_y, \pi\right);$$

$$T_{\sigma_y} \leftarrow \text{BVerify}\left(VK_x, PP, \sigma_y, \pi\right).$$

2.2.2. Definition 4 (Soundness)

Let $\Pi_{\mathcal{A},F}$ *denotes the probability that adversary* $\mathcal{A}$ *succeeds in the Algorithm 1(i.e.* $\Pi_{\mathcal{A},F} = \Pr\left(T_{\sigma_y} \neq \left(reject, S\right) \wedge y \neq F\left(x\right)\right)$*).*

A publicly verifiable computation protocol for a family of functions $\mathcal{F}$ *is sound, if for any adversary* $\mathcal{A}$ *and any function* $F \in \mathcal{F}$, $\Pi_{\mathcal{A},F} \leq \varepsilon$ *and* ε *is a negligible function on the security parameter.*

In a nutshell, a publicly verifiable computation protocol is said to be sound, if for any function $F \in \mathcal{F}$ and any $x \in$ Dom(F), the server S cannot convince any verifier to accept an incorrect result.

3. PUBLICLY VERIFIABLE POLYNOMIAL EVALUATION

3.1. Protocol Overview

This protocol leverages Euclidean division of polynomials as a basis. In detail, for any polynomial F with degree d and polynomial d with degree 2, the Euclidean division of F by B output an unique pair of polynomial $\left(Q, R\right)$ such that $F = BQ + R$, where Q has degree $d-2$ and R has degree ≤ 1.

Consider a resource-constrained client has a polynomial $F\left(X\right)$ of degree d, and he wishes to delegate the evaluation of F to a cloud server S for arbitrary inputs. In order to ensure the privacy of F and computational results, the client executes a disguise process: first, defining random polynomial $B\left(X\right) = X^2 + b_0$, and dividing of F by B output a pair $\left(Q, R\right)$ such that $F = BQ + R$, where $Q\left(X\right) = \sum_{i=0}^{d-2} q_i X^i$ and $R\left(X\right) = r_1 X + r_0$; next, defining another random polynomial $B'\left(X\right) = X^2 + b'_0$, divides of Q by B' output $Q'\left(X\right) = \sum_{i=0}^{d-4} q'_i X^i$ and $R'\left(X\right) =$ where $Q = B'Q' + R'$.

Then the client publishes the public $PK_F = \left(p_1, p_2, p_3\right) = \left(g^{b'_0}, h^{r'_1}, h^{r'_0}\right)$, shares secretly a retrieval key $RK_F = \left(b_0, r_1, r_0\right)$ and sends $EK_F = \left(Q, W\right)$ to the server S, where

$$W = \left(W_0, W_1, \ldots, W_{d-4}\right) = \left(h^{q'_0}, h^{q'_1}, \ldots, h^{q'_{d-4}}\right).$$

If a client wants to evaluate polynomial F at point x, it first sends x to the server S, then computes and publishes the verification key $VK_x = \left(VK_1, VK_2\right) = \left(p_1 g^{x^2}, p_2^x p_3\right)$. After that, the server returns computational result $\sigma_y = Q\left(x\right)$ and the proof $\pi = h^{Q'(x)}$, any verifier can verify the correctness of σ_y by checking whether

$$e\left(g, h^{\sigma_y}\right) = e\left(VK_1, \pi\right) e\left(g, VK_2\right).$$

Finally, the clients rely on RK_F to retrieve the actual results $F\left(x\right) = \left(x^2 + b_0\right)\sigma_y + r_1 x + r_0$.

It is clear that the efficiency of the verification stems from the fact that B' and R' are small degree polynomials. In fact, compared with $O\left(d\right)$ operations to evaluate polynomial F, the client verifies the correctness of σ_y is a small and constant number of computations.

3.2. The Protocol

Assume a client has a d-degree polynomial $F\left(X\right) = \sum_{i=0}^{d} f_i X^i$ with coefficients $f_i \in \mathbb{F}_p$, where p is a large prime. The client would like to outsource the evaluation of F to a cloud service provider S for arbitrary input x. The detail description of our protocol as follows.

- $\mathrm{KeyGen}\left(F, 1^\lambda\right)$. The algorithm first selects three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of the same prime order p, equipped with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Then it picks a generator g_1 and g_2 of groups $\mathbb{G}_1$ and $\mathbb{G}_1$ respectively and defines the public parameters as:

$$PP = \left(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\right)$$

Then selecting a random $b_0 \xleftarrow{R} \mathbb{F}_p^*$ to define a polynomial $B\left(X\right) = X^2 + b_0$, which does not divide polynomial F. Next, performing the Euclidean division of F by B in $\mathbb{F}_P\left[X\right]$. We denote with $Q\left(X\right) = \sum_{i=0}^{d-2} q_i X^i$ the resulting quotient polynomial and denote the resulting remainder polynomial with $R\left(X\right) = r_1 X + r_0$. Selecting another random $b'_0 \xleftarrow{R} \mathbb{F}_p^*$ to define another polynomial $B'\left(X\right) = X^2 + b'_0$, which does not divide polynomial Q. Computing the Euclidean division of Q by B'. Similarly, $Q'\left(X\right) = \sum_{i=0}^{d-4} q'_i X^i$ and $R'\left(X\right) = r'_1 X + r'_0$. The public key is defined by:

$$PK_F = \left(p_1, p_2, p_3\right) = \left(g_1^{b'_0}, g_2^{r'_1}, g_2^{r'_0}\right).$$

The retrieval key $RK_F = \left(b_0, r_1, r_0\right)$. To compute $W_i = g_2^{q'_i} \in \mathbb{G}_2$ for all $0 \le i \le d-4$ and let

$$W = \left(W_0, W_1, \ldots, W_{d-4}\right) \in \mathbb{G}_2^{d-3},\ EK_F = \left(Q, W\right).$$

Algorithm KeyGen concludes its execution by outputting the tuple:

$$\left(PP, PK_F, RK_F, EK_F\right).$$

- $\text{ProbGen}\left(PK_F, PP, x\right)$. Compute $VK_1 = p_1 g_1^{x^2}$, $VK_2 = p_2^x p_3$, then output $\sigma_x = x$ and $VK_x = \left(VK_1, VK_2\right)$.
- $\text{Compute}\left(EK_F, \sigma_x\right)$. Evaluate $\sigma_y = Q\left(x\right) = \sum_{i=0}^{d-2} q_i x^i \mod p$, the proof $\pi = \Pi_{i=0}^{d-4} W_i^{x_i}$ and return the tuple $\left(\sigma_y, \pi\right)$.
- $\text{BVerify}\left(VK_x, PP, \sigma_y, \pi\right)$. If $e\left(g_1, g_2^{\sigma_y}\right) = e\left(VK_1, \pi\right) e\left(g_1, VK_2\right)$, output $T_{\sigma_y} = \left(accept, S\right)$, otherwise output $T_{\sigma_y} = \left(reject, S\right)$.
- $\text{Retrieve}\left(RK_F, T_{\sigma_y}, \sigma_y\right)$. If $T_{\sigma_y} = \left(accept, S\right)$,the clients holding RK_F to retrieve the actual result $y = F\left(x\right) = \left(x^2 + b_0\right)\sigma_y + r_1 x + r_0$. Otherwise returning an error $\perp$.

Some remarks about above protocol are in order.

Firstly, this protocol can ensure the privacy of actual results and the function F. This is an important property in many scenarios.

Secondly, the blind verification algorithm BVerify enables the results can be publicly verified without learning the value.

Thirdly, the disguise process, the blind verification and retrieve actual results can be executed efficiently.

3.3. Security Analysis

Theorem 1. This publicly verifiable polynomial evaluation protocol is correct

Proof. If given $\sigma_x = x \in \mathbb{F}_p$, the server executes algorithm Compute honestly, then the output will correspond to $\left(\sigma_y, \pi\right) = \left(Q\left(x\right), g_2^{Q'(x)}\right)$.This is due to:

$$\pi = \prod_{i=0}^{d-4} W_i^{x^i} = \prod_{i=0}^{d-4} g_2^{q'_i x^i} = g_2^{Q'(x)}.$$

Given that $Q = B'Q' + R'$ in $\mathbb{F}_P\left[X\right]$, we get:

$$e\left(g_1, g_2\right)^{Q(x)} = e\left(g_1, g_2\right)^{B'(x)Q'(x)+R'(x)} = e\left(g_1, g_2^{Q'(x)}\right)^{B'(x)} e\left(g_1, g_2\right)^{R'(x)}.$$

Since $VK_1 = p_1 g_1^{x^2} = g_1^{x^2+b'_0} = g_1^{B'(x)}$ and $VK_2 = p_2^x p_3 = g_2^{r'_1 x + r'_0} = g_2^{R'(x)}$.

From it the authors conclude that $e\left(g_1, g_2^{\sigma_y}\right) = e\left(VK_1, \pi\right) e\left(g_1, VK_2\right)$.

Then $\text{BVerify}\left(VK_x, PP, \sigma_y, \pi\right) \rightarrow \left(accept, S\right)$. It is clear that the clients holding RK_F can retrieve the actual result by computing $y = F\left(x\right) = \left(x^2 + b_0\right)\sigma_y + r_1 x + r_0$.

Theorem 2. The protocol proposed above for publicly verifiable polynomial evaluation is sound under the $\left[d/2-1\right]$ - SDH assumption

Proof. Assume an adversary $\mathcal{A}$ can break the soundness of the protocol with a non-negligible advantage ξ. In the following, the authors demonstrate there exists another adversary $\mathcal{B}$ that breaks the $\left[d/2-1\right]$ SDH assumption with a non-negligible advantage $\delta \geq \xi$.

The authors denote with $\mathcal{O}_{sdh}$ an oracle, which returns a pair $\left(g_1, g_1^{\alpha}\right)$ in $\mathbb{G}_1$ and a tuple $\left(g_2, g_2^{\alpha}, \ldots, g_2^{\alpha^{\lceil d/2-1 \rceil}}\right)$ in $\mathbb{G}_{2,}$ for $\alpha \xleftarrow{R} \mathbb{F}_p^*$.

Firstly, the adversary $\mathcal{B}$ calls oracle $\mathcal{O}_{sdh}$ to obtain a tuple

$$\left(g_1, g_1^{\alpha}, g_2, g_2^{\alpha}, \ldots, g_2^{\alpha^{\lceil d/2-1 \rceil}}\right),$$

then simulates the soundness experiment (detail in Algorithm 1) to adversary $\mathcal{A}$. That is when $\mathcal{A}$ calls oracle $\mathcal{O}_{\text{KeyGen}}$ with polynomial $Q\left(X\right) = \sum_{i=0}^{d-2} q_i X^i$ in $\mathbb{F}_P\left[X\right]$, adversary $\mathcal{B}$ simulate response $\mathcal{O}_{\text{KeyGen}}$ as follows:

1. Define the public parameters $\widehat{PP} = \left(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\right)$;
2. It defines the evaluation $\widehat{EK}_F = \left(Q, \widehat{W}\right)$, where $\widehat{W} = \left(\widehat{W}_0, \widehat{W}_1, \ldots, \widehat{W}_{d-4}\right)$. Let $\widehat{W}_{d-4} = g_2^{q_{d-2}}$ and $\widehat{W}_{d-5} = g_2^{q_{d-3}}$. For each $2 \le k \le d-4$, it computes $\widehat{W}_{d-4-k} = \Pi_{i=0}^{\lceil k/2 \rceil} g_2^{q_{d-2-k+2i}\left(-1\right)^i \alpha^i}$.
3. Computing the public key $\widehat{PK}_F = \left(\hat{p}_1, \hat{p}_2, \hat{p}_3\right)$ as following:

$$\hat{p}_1 = g_1^{\alpha},$$
$$\hat{p}_2 = \prod_{i=0}^{\lceil (d-3)/2 \rceil} g_2^{q_{2i+1}\left(-1\right)^i \alpha^i},$$
$$\hat{p}_3 = \prod_{i=0}^{\lceil d/2-1 \rceil} g_2^{q_{2i}\left(-1\right)^i \alpha^i}.$$

If $\left(\hat{p}_2, \hat{p}_3\right) = \left(1,1\right)$, then adversary $\mathcal{B}$ stops the experiment.

4. Otherwise, it return $\left(\widehat{PP}, \widehat{PK}_F, \widehat{EK}_F\right)$ to adversary $\mathcal{A}$.

The tuple $\left(\widehat{PP}, \widehat{PK}_F, \widehat{EK}_F\right)$ returned by adversary $\mathcal{B}$ is statistically indistinguishable from the distribution of $\left(PP, PK_F, EK_F\right)$ in the soundness experiment. In fact, if they denote for all $0 \le i \le d-4$, $\widehat{W}_i = g_2^{q'_i}$ and let $\left(\hat{p}_2, \hat{p}_3\right) = \left(g_2^{r'_1}, g_2^{r'_0}\right)$, they can verify that:

- $q_{d-2} = q'_{d-4} \bmod p$ and $q_{d-3} = q'_{d-5} \bmod p$;
- For all $2 \le i \le d-4$, $q_i = \alpha q'_i + q'_{i-2} \bmod p$;
- $q_1 = \alpha q'_1 + r'_1 \bmod p$ and $q_0 = \alpha q'_0 + r'_0 \bmod p$;
- $\left(r'_1, r'_0\right) \ne \left(0,0\right)$.

Next, the authors define $Q'(X) = \sum_{i=0}^{d-4} q'_i X^i$, $B'(X) = X^2 + \alpha$, $R'(X) = r'_1 X + r'_0$, the following equation is correct: $Q = B'Q' + R'$.

The authors can conclude (i) B' does not divide Q; (ii) each $\widehat{W}_i$ correctly encodes the ith coefficient of Q'; (iii) the pair $\left(\hat{p}_2, \hat{p}_3\right)$ correctly encodes R'.

Then, adversary $\mathcal{A}$ selects a challenge value $x \in \mathbb{F}_p$ and calls oracle $\mathcal{O}_{\text{ProbGen}}$ with the tuple $\left(\widehat{PK}_F, \widehat{PP}, x\right)$. Accordingly, adversary $\mathcal{B}$ compute the response of oracle $\mathcal{O}_{\text{ProbGen}}$ and returns verification key $VK_x = \left(VK_1, VK_2\right) = \left(\hat{p}_1 g_1^{x^2}, \hat{p}_2^x \hat{p}_3\right)$.

Finally, adversary $\mathcal{A}$ returns a pair $\left(\sigma_y, \pi\right)$, where $\sigma_y \neq Q(X)$ and $\left(\sigma_y, \pi\right)$ is accepted by algorithm BVerify with a non-negligible advantage ξ.

Hence, adversary $\mathcal{B}$ breaks the $\left[d/2-1\right]$ - SDH assumption by first computing $Q(X)$ and $\pi^* = \prod_{i=0}^{d-4} \widehat{W}_i^{x^i}$, and outputting:

$$\left(\beta, g_2^{1/(\beta+\alpha)}\right) = \left(x^2, \left(\frac{\pi}{\pi^*}\right)^{\left(y-Q(x)\right)^{-1}}\right).$$

Indeed, since the pair $\left(\sigma_y, \pi\right)$ satisfies the verification,

$$e\left(g_1, g_2^{\sigma_y}\right) = e\left(\hat{p}_1 g_1^{x^2}, \pi\right) e\left(g_1, \hat{p}_2^x \hat{p}_3\right) = e\left(g_1^{x^2+\alpha}, \pi\right) e\left(g_1, \hat{p}_2^x \hat{p}_3\right) \tag{1}$$

Furthermore,

$$e\left(g_1, g_2^{Q(x)}\right) = e\left(g_1^{x^2+\alpha}, \pi^*\right) e\left(g_1, \hat{p}_2^x \hat{p}_3\right) \tag{2}$$

By dividing Equation (1) by (2), the authors obtain:

$$e\left(g_1, g_2\right) = e\left(g_1^{x^2+\alpha}, \left(\frac{\pi}{\pi^*}\right)^{\left(\sigma_y - Q(x)\right)^{-1}}\right).$$

Hence if adversary $\mathcal{B}$ does not stop the experiment, then it will be able to break the $\left[d/2-1\right]$ - SDH assumption. If adversary $\mathcal{B}$ aborts the experiment when $\left(\hat{p}_2, \hat{p}_3\right) = (1,1)$, then the adversary $\mathcal{B}$ can conclude B' divides Q. This means adversary $\mathcal{B}$ can find α, that break the $\left[d/2-1\right]$ - SDH assumption obviously.

Thus, the authors deduce that if there exists an adversary $\mathcal{A}$ that can break the soundness of the protocol with a non-negligible advantage ξ . They demonstrated there exists another adversary $\mathcal{B}$ that breaks the $\left[d/2-1\right]$ - SDH assumption with a non-negligible advantage $\delta\delta \geq \xi$.

3.4. Performance Analysis

Algorithm KeyGen first selects two random coefficients b_0 , b'_0 and executes two Euclidean divisions. The latter operation consists of $d+\left(d-2\right)$ multiplications and additions. Next, it needs to performs one exponentiation in $\mathbb{G}_1$ and $d-1$ exponentiations in $\mathbb{G}_2$ to construct the PK_F and the EK_F . Although the algorithm KeyGen computationally expensive, it is executed only once. Besides, its computational cost can be amortized.

Algorithm ProbGen generates the verification key $VK_x = \left(VK_1, VK_2\right)$, which costs a constant number of operations. In particularly, these operations do not depend on the degree of polynomial. In detail, this algorithm's work consists of computing x^2 in $\mathbb{F}_p$, executing one exponentiation and one multiplication in $\mathbb{G}_1$, and performing one exponentiation and one multiplication in $\mathbb{G}_2$.

Algorithm Compute evaluate the result $\sigma_y = Q\left(x\right) = \sum_{i=0}^{d-2} q_i x^i \mod p$ and the proof $\pi = \Pi_{i=0}^{d-4} W_i^{x_i}$. In detail, it demands $2d-5$ multiplications in $\mathbb{F}_p$, $d-3$ exponentiations and $d-4$ multiplications in $\mathbb{G}_2$.

Algorithm BVerify demands one exponentiation and one division in $\mathbb{G}_2$ and the computation of 2 bilinear pairings.

Finally, the algorithm Retrieve only demands 6 additions and multiplications in $\mathbb{F}_p$.

In regard to storage, the client needs to keep the public key and the retrieval key. Meanwhile, the server stores the coefficients of polynomial Q and the encodings W_i . (See Table 2)

Table 2. This is a summary of the efficiency of our protocol

Algorithm	Efficiency	Client's storage	Server's storage
KeyGen	2 prng and $2d-2$ mul in F_p 1 exp in G_1 $d-1$ exp in G_2	O_1	O_{d-2}
ProbGen	1 mul in F_p 1 exp and 1 mul in G_1 1 exp and 1 mul in G_2	-	-
Compute	$2d-5$ mul in F_p $d-3$ exp and $d-4$ mul in G_2	-	-
BVerify	1 exp and 1 div in G_2 2 pairings	-	-
Retrieve	6 mul in F_p	-	-

4. CONCLUSION

In this paper, the authors propose a privacy-preserving and publicly verifiable protocol for outsourcing polynomials evaluation to a malicious cloud. The protocol based on Euclidean division properties. Compared with existing protocols, it can ensure the privacy of functions and actual results, and it supports the blind verifiability. Besides, the disguise process, the blind verification and the retrieve can be executed efficiently.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No.61772311.

REFERENCES

Babai, L. (1985). Trading group theory for randomness. In *ACM Symposium on Theory of Computing*, Providence, Rhode Island, May 6-8 (pp. 421-429). DBLP.

Backes, M., Fiore, D., & Reischuk, R. M. (2013). Verifiable delegation of computation on outsourced data. *ACM Sigsac Conference on Computer & Communications Security* (Vol.37, pp.863-874). ACM. doi:10.1145/2508859.2516681

Benabbas, S., Gennaro, R., & Vahlis, Y. (2011). Verifiable delegation of computation over large datasets. In *Conference on Advances in Cryptology* (Vol. 52, pp. 111-131). Springer-Verlag. doi:10.1007/978-3-642-22792-9_7

Blanton, M., Zhang, Y., & Frikken, K. B. (2013). Secure and verifiable outsourcing of large-scale biometric computations. *ACM Transactions on Information and System Security*, *16*(3), 1–33. doi:10.1145/2535523

Catalano, D., & Fiore, D. (2014). Practical Homomorphic MACs for Arithmetic Circuits. *International Conference on Public-Key Cryptography - PKC* (Vol. 7881, pp. 538-555). Springer-Verlag Inc.

Elkhiyaoui, K., Önen, M., Azraoui, M., & Molva, R. (2016, May). Efficient techniques for publicly verifiable delegation of computation. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (pp. 119-128). ACM.

Fiore, D., & Gennaro, R. (2012). Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *ACM Conference on Computer and Communications Security* (pp.501-512). ACM. doi:10.1145/2382196.2382250

Gennaro, R., Gentry, C., & Parno, B. (2010). Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Conference on Advances in Cryptology* (Vol. 6223, pp.465-482). Springer-Verlag. doi:10.1007/978-3-642-14623-7_25

Goldwasser, M., Micali, S., & Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, *18*(1), 186–208. doi:10.1137/0218012

Hu, X., & Tang, C. (2015). Secure outsourced computation of the characteristic polynomial and eigenvalues of matrix. *Journal of Cloud Computing*, *4*(1), 7. doi:10.1186/s13677-015-0033-9

Kilian, J. (1995). Improved Efficient Arguments (Preliminary Version). In *International Cryptology Conference on Advances in Cryptology* (Vol. 963, pp.311-324). Springer-Verlag.

Kilian, J. (1998). A note on efficient zero-knowledge proofs and arguments.

Lei, X., Liao, X., Huang, T., & Li, H. (2015). Cloud computing service: The case of large matrix determinant computation. *IEEE Transactions on Services Computing*, *8*(5), 688–700. doi:10.1109/TSC.2014.2331694

Liu, Y., Wang, X. A., Sangaiah, A. K., & Shao, H. (2018). Publicly verifiable 1-norm and 2-norm operations over outsourced data stream under single-key setting.

Luo, X., Yang, X., Cong, L. I., & Wang, X. (2018). Publicly verifiable outsourced computation scheme for multivariate polynomial based on two-server model.

Papamanthou, C., Shi, E., & Tamassia, R. (2013). Signatures of correct computation. In *Theory of Cryptography Conference on Theory of Cryptography* (Vol. 7785, pp. 222-242). Springer-Verlag. doi:10.1007/978-3-642-36594-2_13

Parno, B., Raykova, M., & Vaikuntanathan, V. (2012). *How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption. In Theory of Cryptography*. Springer Berlin Heidelberg.

Song, W., Wang, B., Wang, Q., Shi, C., Lou, W., & Peng, Z. (2017). Publicly verifiable computation of polynomials over outsourced data with multiple sources. *IEEE Transactions on Information Forensics & Security*.

Sun, J., Zhu, B., Qin, J., Hu, J., & Wu, Q. (2018). Confidentiality-preserving publicly verifiable computation. *International Journal of Foundations of Computer Science*, *28*(6), 799–818. doi:10.1142/S0129054117400196

Wang, Q., Zhou, F., Chen, C., Xuan, P., & Wu, Q. (2017). Secure collaborative publicly verifiable computation. *IEEE Access: Practical Innovations, Open Solutions*, *5*(99), 2479–2488. doi:10.1109/ACCESS.2017.2672866

Ye, J., Zhang, H., & Fu, C. (2016). Verifiable delegation of polynomials. *International Journal of Network Security*, *18*(2), 283–290.

Zhang, K., Wei, L., Li, X., & Qian, H. (2017). *Provably Secure Dual-Mode Publicly Verifiable Computation Protocol in Marine Wireless Sensor Networks*. Wireless Algorithms, Systems, and Applications. doi:10.1007/978-3-319-60033-8_19

Zhang, L. F., & Safavi-Naini, R. (2014). Verifiable Delegation of Computations with Storage-Verification Trade-off. In *European Symposium on Research in Computer Security* (Vol. 8712, pp.112-129). Springer. doi:10.1007/978-3-319-11203-9_7

Zhang, X., Jiang, T., Li, K. C., & Chen, X. (2017). New publicly verifiable computation for batch matrix multiplication. *Information Sciences*. doi:10.1016/j.ins.2017.11.063

Zhou, L., & Li, C. (2017). Outsourcing eigen-decomposition and singular value decom- position of large matrix to a public cloud. *IEEE Access: Practical Innovations, Open Solutions*, *4*, 869–879. doi:10.1109/ACCESS.2016.2535103

Dawei Xie is studying in the school of mathematics, Shandong University.

Haining Yang received the B.S. degree in School of Mathematics and Statistics from Shandong Normal University, China, in 2016. He is currently a Ph.D. candidate in School of Mathematics of Shandong University, China. His research interests include cloud security and cryptography.

Qin jing is a professor at School of Mathematics, Shandong University. Subject is basic mathematics, research direction is cryptography. In recent years, the research mainly focuses on public key searchable encryption, verifiable outsourcing calculation, data integrity detection, computational complexity theory, attribute-based encryption, casual transmission, secure multi-party calculation and other aspects. She was a visiting scholar at the university of Pennsylvania from October 2010 to May 2011 and a coresearcher at the university of Greenwich from June to July 2018. She has presided over and completed 3 projects of the national natural science foundation of China and 3 projects of the provincial natural science foundation of China, 1 project of the national "11th five-year" theoretical code foundation of China, 1 project of the Chinese postdoctoral foundation of China, and 2 projects of the national key laboratory opening project.