Hindawi
Security and Communication Networks
Volume 2018, Article ID 5275132, 15 pages
https://doi.org/10.1155/2018/5275132

Research Article

Confidentiality-Preserving Publicly Verifiable Computation Schemes for Polynomial Evaluation and Matrix-Vector Multiplication

Jiameng Sun,[1] Binrui Zhu,[1] Jing Qin,[1,2] Jiankun Hu,[3] and Jixin Ma[4]

[1]*School of Mathematics, Shandong University, Jinan, Shandong 250100, China*
[2]*State Key Laboratory of Information Security Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[3]*School of Engineering and Information Technology, University of New South Wales Defence Force Academy, Canberra, Australia*
[4]*Centre for Computer and Computational Science, School of Computing and Mathematical Sciences, University of Greenwich, London, UK*

Correspondence should be addressed to Jing Qin; qinjing@sdu.edu.cn

Received 4 December 2017; Revised 9 May 2018; Accepted 26 May 2018; Published 21 June 2018

Academic Editor: Mamoun Alazab

With the development of cloud services, outsourcing computation tasks to a commercial cloud server has drawn attention of various communities, especially in the Big Data era. Public verifiability offers a flexible functionality in real circumstance where the cloud service provider (CSP) may be untrusted or some malicious users may slander the CSP on purpose. However, sometimes the computational result is sensitive and is supposed to remain undisclosed in the public verification phase, while existing works on publicly verifiable computation (PVC) fail to achieve this requirement. In this paper, we highlight the property of result confidentiality in publicly verifiable computation and present confidentiality-preserving public verifiable computation (CP-PVC) schemes for multivariate polynomial evaluation and matrix-vector multiplication, respectively. The proposed schemes work efficiently under the amortized model and, compared with previous PVC schemes for these computations, achieve confidentiality of computational results, while maintaining the property of public verifiability. The proposed schemes proved to be secure, efficient, and result-confidential. In addition, we provide the algorithms and experimental simulation to show the performance of the proposed schemes, which indicates that our proposal is also acceptable in practice.

1. Introduction

Outsourcing computation has been served as a significant service with the rapid development of Cloud Computing Technology. It provides the service purchaser (whom we call user) with constraint computational power to delegate the complicated computational tasks to the service provider (which we call cloud server) and enjoy its unlimited computational resources in a pay-per-use manner. This brings a huge convenience for resource-constraint devices to reduce their computational overhead and thus has attracted significant interests in both industrial and academic communities. A number of large enterprise groups, such as Amazon, Google, and Alibaba, have launched their Cloud Computing to provide computation outsourcing services. What is more, in Big Data era, the ability to deal with the massive data has become core competitiveness while outsourcing computation just fits this demand.

While outsourcing computation paradigm enjoys numerous benefits, it also suffers from rigorous challenges [1]. To begin with, since the cloud server is commercialized, sometimes it may not perform the computation honestly but output a computationally indistinguishable result in order to save its cost for more interests. Therefore, a basic requirement of outsourcing computation is to assure the correctness of the computational result. In other words, the user should have a way to verify the correctness of the output from the cloud server with an overwhelming probability. Despite the untrustworthy of cloud server, misbehavior may also happen from the user side. For example, a malicious user may deliberately claim the output from the cloud server incorrect and slander the cloud service provider by this even if the cloud server has performed the computation honestly. This is due to the fact that verification is done in a private manner. Therefore, it is preferable that the verification can be done publicly. That is to say, anyone except the user himself is able to verify the output from the cloud server. With public verification, not only cannot the cloud server cheat with an incorrect output, but also the user cannot claim the output from the cloud server incorrect for no reason, because now the output is witnessed and verified by everyone. Secondly, since sometimes the computational result is something sensitive, it needs to be kept secret to any party except the user himself. Thus another challenge of outsourcing computation is to assure confidentiality of the computational result, especially when the output from the cloud server can be verified publicly. Last but not least, the whole workload of the user in certain computation outsourcing procedure must be much less than accomplishing this computation task all by the user himself. We call this the requirement of efficiency. This is essential because if not, the outsourcing will be meaningless.

The evaluation of multivariate polynomials $p(x)$ is one of the most fundamental computational tasks in scientific communities. In practice, there are so many problems that can be reduced to a model of evaluating certain polynomial with multivariate input value, for example, to evaluate an employee's performance in a company and to evaluate a person's health condition. Matrix-vector multiplication is another fundamental computational task that is widely applied, for example the Discrete Fourier Transform (DFT) and the Singular Value Decomposition (SVD). And in Big Data era, with the data we need to deal with getting more and more enormous, it is very likely that the storage requirement when evaluating multivariate polynomials or matrix-vector multiplication exceeds the available memory of the user's computational devices, like cell phones or portable laptop. Thus we need to find another way to fulfill the computational tasks securely and efficiently. Plenty of works have been done to seek secure and efficient schemes of outsourcing computation for polynomials and matrix-vector multiplication. Fiore and Gennaro [2] proposed schemes to securely outsource evaluation of multivariate polynomials and matrix-vector multiplication and verify the corresponding result in a public manner. Unfortunately, one disadvantage of their proposal is the leakage of final result. Anyone is able to verify the correctness of the output from the cloud server and then obtain the result of the target evaluation. This brings a drawback in practice when the result is something sensitive, for example, the year-end bonus of an employee and the health condition of a person.

1.1. Related Works

1.1.1. Verifiable Computation. Verifiable computation (VC) was first proposed by Gennaro et al. [3]. In VC, only two parties are involved, the client that processes the input data and the server that evaluates the target function with the value client sends. The output of the server can be verified by the client only. Both the input and output value of the function are private in the whole procedure. Gennaro et al. proposed a concrete VC scheme for arbitrary circuit using Yao's [4] two-party computation scheme and Gentry's [5] fully homomorphic encryption (FHE) scheme. After that, different VC schemes using FHE were proposed [6–8]. They made use of various techniques to achieve verifiability. However, applying FHE in practice brings expensive overhead.

To avoid applying FHE and promote efficiency, a lot of papers [9–15] focused on VC schemes for various kinds of computation and had achieved outstanding results.

1.1.2. Publicly Verifiable Computation. Different from VC where the verification is done privately, publicly verifiable computation (PVC) allows anyone to verify the result output by the server. PVC brings more flexible application in the untrusted cloud environment than VC, mainly in two-fold. One is to release the workloads of verification for the client. Another is the supervision of users. This is because if the verification is only done privately, once an incorrect result is claimed, it is hard to tell whether the server misbehaved or the user is intentionally slandering. However if the result can be verified publicly, the user's slandering is easy to detect. PVC was first proposed by Parno et al. [16]. They constructed a PVC scheme for Boolean functions using KP-ABE schemes. After that, many PVC schemes were proposed [2, 17–19]. In 2012, Fiore et al. [2] proposed a PVC scheme for multivariate polynomial evaluation and matrix multiplication. They took inspiration from Benabbas et al.'s [9] VC scheme to use pseudorandom function that enjoys closed form efficiency to generate the verification key efficiently, and they generalized the function to multivariate case. Moreover, by leveraging the technique of bilinear map, they have improved the verification procedure from private to public manner. With the similar technique, Sun et al. [17] constructed batch verifiable computation schemes with public verification for polynomial and matrix that achieve simultaneously evaluation of multiple functions in one outsourcing phase. In 2016, Elkhiyaoui et al. proposed another solution for univariate polynomial evaluation and matrix multiplication. They leverage the idea of Euclidean division of polynomials to construct the structure of the verifiable computation. And the bilinear map technique is utilized to make the verification able to be public. However, this idea is only suitable for univariate polynomial scenario. What is more, all the schemes mentioned above share the same disadvantage that anyone except for the user that verifies the result will surely obtain its concrete value. This is insecure in practice when the result of certain computation is usually something sensitive and the verification process is supposed to output a judgement to the correctness of the result rather than disclosing the value itself. To overcome this, Alderman et al. [20] improved Parno et al.'s [16] scheme with a secret substitution bit and presented a PVC with key distribution center (KDC) for Boolean functions. They also achieved other properties like revocation based on this PVC with KDC [21, 22], but this way of using secret substitution bit cannot fit other functions that have large range.

1.2. Our Contributions. In this paper, we present a modified PVC model that is considered to be more practical. It captures the confidentiality of the computational result, which we believe is an important property when utilizing in practice.

We present outsourcing schemes for securely and efficiently evaluating high degree multivariate polynomials and matrix-vector multiplication. Compared with existing outsourcing schemes for polynomials [2, 23] and matrix multiplication, our proposal simultaneously captures properties of both public verifiability and result confidentiality. This offers a more flexible application in practice.

We also provide the algorithm for our outsourcing scheme and run some simulated experiments to show the efficiency of the proposed schemes.

This paper is an extension of its corresponding conference version [24]. In the revised version, we extends the CP-PVC scheme to the matrix-vector multiplication case. The resulting scheme proved to achieve not only the properties of polynomial case (i.e., security, public verifiability, and result confidentiality) but also the input privacy. We run corresponding simulated experiments and the result is also acceptable for practice.

1.3. Paper Organization. The remaining parts of the paper are organized as follows. Some necessary preliminaries are provided for the proposed schemes in Section 2. The framework of the proposed CP-PVC protocol is defined in Section 3. The concrete constructions of the CP-PVC schemes for polynomial evaluation and matrix-vector multiplication are presented, analyzed, and simultaneously experimented separately in Sections 4 and 5. The conclusion of the paper is in Section 6.

2. Preliminaries

In this section, we provide some definitions about algebraic pseudorandom function (PRF) with closed form efficiency, bilinear Map, and some related notions. We also provide the computational assumptions that are used for the construction of our schemes.

2.1. Algebraic PRF with Closed Form Efficiency. One of our main techniques is the PRF with closed form efficiency. PRF is a function (denoted by R) that is generated from a secret seed K. It owns the properties of both randomness and computational efficiency. A closed form efficient PRF consists of algorithms ($\mathbf{KG}$, R) that are defined as follows:

(i) $\mathbf{KG}(1^\lambda) \rightarrow (K, pp)$: The randomized key generation algorithm takes as input the security parameter 1^λ and outputs a tuple of parameters (K, pp), where K denotes the secret seed and pp denotes the public parameters that specifies the domain X and range Y of the function, respectively.

(ii) $R(K, x) \rightarrow y$: The deterministic functional computation algorithm takes as input the secret seed K and the value $x \in X$ and computes a value $y \in Y$. We usually denote it by $R_K(\cdot)$.

An algebraic PRF with closed form efficiency must satisfies the following properties:

(1) (*Algebraic*) A PRF $(\mathbf{KG}, R)$ is algebraic if the range Y of $R_K(\cdot)$ forms an abelian group. We use multiplication notation for group operation.

(2) (*Pseudorandom*) A PRF $(\mathbf{KG}, R)$ is pseudorandom if for every PPT adversary $\mathscr{A}$, there holds

$$\left|\Pr\left[\mathscr{A}^{R_K(\cdot)}\left(1^\lambda, pp\right) = 1\right] - \Pr\left[\mathscr{A}^{r(\cdot)}\left(1^\lambda, pp\right) = 1\right]\right| \leq \epsilon \quad (1)$$

where $(K, pp) \overset{\$}{\leftarrow} \mathbf{KG}(1^\lambda)$ and $r : X \rightarrow Y$ is a random function.

(3) (Closed form efficiency) Let $\mathbf{Comp}$ represent arbitrary computation that takes as input l random values $r_1, \ldots, r_l \in Y$ and a vector of m arbitrary values $x = (x_1, \ldots, x_m)$. Assume that the fastest algorithm to compute $\mathbf{Comp}(r_1, \ldots, r_l, x_1, \ldots, x_m)$ takes time t_0. Let $z = (z_1, \ldots, z_l)$ be a tuple of arbitrary values taken from X. Then a PRF $(\mathbf{KG}, R)$ is closed form efficient for $(\mathbf{Comp}, z)$ if there exists an algorithm $\mathbf{CFEval}_{\mathbf{Comp},z}$ such that

$$\begin{aligned}&\mathbf{CFEval}_{\mathbf{Comp},z}(K, x)\\&\quad = \mathbf{Comp}\left(R_K(z_1), \ldots, R_K(z_l), x_1, \ldots, x_m\right)\end{aligned} \quad (2)$$

and its running time is $o(t_0)$. When $z = (1, \ldots, l)$, we usually omit it from the subscript and write $\mathbf{CFEval}_{\mathbf{Comp}}(K, x)$ instead.

Here we only show the definition of PRF with closed form efficiency. We will give a concrete algorithm of PRF with closed form efficiency for multivariate polynomials in Section 3.

2.2. Bilinear Map. Our constructions also use bilinear maps. Bilinear pairing is a powerful tool in noninteractive authentication and has been widely applied in both encryption and signature schemes [9, 25]. To be specific, let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be finite cyclic multiplicative groups of order p, and let g, h be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. A map $\widehat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is called a bilinear map if it satisfies the following properties:

(1) Bilinearity: it holds that

$$\widehat{e}\left(g^\alpha, h^\beta\right) = \widehat{e}(g, h)^{\alpha\beta} \quad (3)$$

for all $\alpha, \beta \in \mathbb{Z}_N$.

(2) Nondegeneracy: there exist $g_1 \in \mathbb{G}_1, h_1 \in \mathbb{G}_2$ such that $\widehat{e}(g_1, h_1) \neq 1$.

(3) Computability: there exists an efficient algorithm to compute $\widehat{e}(g_1, h_1)$ for any $g_1 \in \mathbb{G}_1, h_1 \in \mathbb{G}_2$.

2.3. Computational Assumptions. The computational assumption that is used for the construction of the PRF with closed form efficiency is decision linear (DL) assumption. We present the definition below.

Definition 1 (decision linear assumption). Let $\mathbb{G}$ be a group of prime order p. Given $g_0, g_1, g_2 \in \mathbb{G}$ and $r_0, r_1, r_2 \in \mathbb{Z}_p$, one defines the advantage of an algorithm $\mathscr{A}$ in deciding the decision linear problem in $\mathbb{G}$ as

$$\begin{aligned}\mathbf{Adv}_{\mathscr{A}}^{dl} = \Big|\Pr\left[\mathscr{A}\left(p, g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, g_0^{r_1+r_2}\right) = 1\right] \\ - \Pr\left[\mathscr{A}\left(p, g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}, g_0^{r_0}\right) = 1\right]\Big|.\end{aligned} \tag{4}$$

One says that the decision linear assumption (t, ϵ) holds in $\mathbb{G}$ if for every t-time algorithm $\mathscr{A}$ one has $\mathbf{Adv}_{\mathscr{A}}^{dl} \le \epsilon$.

Note that the decision linear assumption holds in generic bilinear groups. Relative proof can be found in [26].

Next we present the definition of co-CDH, which is the base for the security of our proposed schemes. The co-CDH assumption was first introduced in BLS signature scheme presented by Boneh et al. [27], as a natural extension of standard CDH problem in asymmetric bilinear pairing. It is defined as follows.

Definition 2 (co-CDH assumption). Let $(p, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ be as above in Section 2.2. Given random $a, b \in \mathbb{Z}_p$, one defines the advantage of an algorithm $\mathscr{A}$ in solving the co-CDH problem in $\mathbb{G}_1, \mathbb{G}_2$ as

$$\mathbf{Adv}_{\mathscr{A}}^{co-cdh} = \left|\Pr\left[\mathscr{A}\left(p, g, h, g^a, h^b\right) = g^{ab}\right]\right| \tag{5}$$

and one says that the co-CDH assumption (t, ϵ) holds in $\mathbb{G}_1, \mathbb{G}_2$ if for every t-time algorithm $\mathscr{A}$ one has $\mathbf{Adv}_{\mathscr{A}}^{co-cdh} \le \epsilon$.

Note that when $\mathbb{G}_1 = \mathbb{G}_2$, the co-CDH problem reduces to standard CDH.

3. Modelling CP-PVC

We use an amortized model [16] to construct our CP-PVC scheme. That is, the user (denoted by SU) shall invest a larger amount of computational work in a preprocessing phase in order to obtain efficiency during the computation outsourcing phase. The adversaries in a PVC protocol are two types, the cloud server (denoted by CS) and some "curious" verifiers (denoted by CV). The former is in lazy-but-honest model [28] and the latter is in honest-but-curious model. This is reasonable since, in practice, a rational commercial cloud service will try to minimize the computation it needs to do to pass the verification algorithm. And passing the verification algorithm is its priority because only in this way can it get the payback. Also since the verification is public, there will be some curious verifiers that perform the public verification algorithm and try to discover some secret information about the final result value.

A difference between the framework of PVC proposed by Parno et al. [16] and ours is that we address the confidentiality of the computational result. In the public verification phase, a bit is output instead of the result value. And the result value is obtained in the later phase called private retrieval. To realize confidentiality, the user needs to operate the target function in the preprocessing phase and obtain a secret key for retrieval and keep it secret.

Let $\mathscr{F}$ be a class of functions and $F \in \mathscr{F}$. We define a confidentiality-preserving publicly verifiable computation (CP-PVC) protocol π via the following five algorithms:

(i) $\mathsf{KeyGen}(1^\lambda, F) \rightarrow (SK_F, EK_F, PP)$: The randomized key generation algorithm takes as input a security parameter and the function F and as outputs a secret key SK_F for the input delegation phase, an evaluation key EK_F for the cloud server to compute the outsourced message, and the public parameter PP. This is done by the client.

(ii) $\mathsf{ProbGen}(PP, SK_F, x) \rightarrow (\sigma_x, VK_x, RK_x)$: Given the public parameter PP, the secret key SK_F, and the input value $x \in \mathsf{Domain}(F)$, the randomized problem generation algorithm outputs a public value σ_x, which is the encoding of x, together with a public verification key VK_x for nonclient parties to verify the correctness and a private retrieval key RK_x for the client to retrieve final result $F(x)$. This is done by the client.

(iii) $\mathsf{Compute}(EK_F, \sigma_x) \rightarrow \sigma_{out}$: On inputting the evaluation key EK_F together with the value σ_x, the randomized computation algorithm outputs a value σ_{out}. This is done by the worker (cloud server).

(iv) $\mathsf{PubVer}(PP, VK_x, \sigma_{out}) \rightarrow \{accept, reject\}$: The deterministic public verification algorithm uses the public parameter PP and public verification key VK_x to check whether the final result is correct and returns *accept* or *reject* accordingly. This is done by the nonclient verifiers.

(v) $\mathsf{PrivRet}(PP, VK_x, RK_x, \sigma_{out}) \rightarrow y$: The deterministic private retrieval algorithm is run on input PP, VK_x,RK_x, and σ_{out} to compute a string $y \in \{0,1\}^* \cup \{\perp\}$. Here, the special symbol $\perp$ indicates that the public verification algorithm rejects the worker's answer σ_{out}. This is done by the client.

A verifiable computation scheme should be both correct and secure. We give the definition of correctness and security in the following.

Definition 3 (correctness). A confidentiality-preserving publicly verifiable computation protocol π is correct for a class of functions $\mathscr{F}$ if, for any F from $\mathscr{F}$, any tuple (SK_F, EK_F, PP) output by $\mathsf{KeyGen}(1^\lambda, F)$, any x chosen from $\mathsf{Domain}(F)$, any tuple (σ_x, VK_x, RK_x) output by $\mathsf{ProbGen}(PP, SK_F, x)$, and any σ_{out} output by $\mathsf{Compute}(EK_F, \sigma_x)$, the PubVer algorithm on input (PP, VK_x, σ_{out}) outputs *accept*, and the $\mathsf{PrivRet}$ algorithm on input PP, VK_x, RK_x, and σ_{out} outputs $y = F(x)$.

The security of a verifiable computation requires that the worker is not able to output an incorrect value that passes the PubVer or the PrivRet algorithm. We give the formal definition via the following experiment.

Definition 4 (security). Let π be a confidentiality-preserving publicly verifiable computation scheme for a class of functions $\mathscr{F}$, and assume that $\mathscr{A}$ is PPT adversaries. Consider Experiment $\mathrm{EXP}_{\mathscr{A}}[\pi, F, \lambda]$ for any $F \in \mathbb{F}$ below:

```
Experiment $EXP_{\mathscr{A}}[\pi, F, \lambda]$
KeyGen$(1^{\lambda}, F) \xrightarrow{\$} (SK_F, EK_F, PP)$;
For $i = 1$ to $q$,
$\mathscr{A}(PP, EK_F, \sigma_{x,1}, VK_{x,1}, \ldots, \sigma_{x,i-1}, VK_{x,i-1}) \rightarrow x_i$;
ProbGen$(PP, SK_F, x_i) \xrightarrow{\$} (\sigma_{x_i}, VK_{x_i}, RK_{x_i})$;
$\mathscr{A}(PP, EK_F, \sigma_{x,1}, VK_{x,1}, \ldots, \sigma_{x,q}, VK_{x,q}) \rightarrow x^*$;
ProbGen$(PP, SK_F, x^*) \xrightarrow{\$} (\sigma_{x^*}, VK_{x^*}, RK_{x^*})$;
$\mathscr{A}(\sigma_{x^*}, VK_{x^*}) \rightarrow (\sigma_{out^*})$;
PubVer$(VK_{x^*}, \sigma_{out^*}) \rightarrow k$
PrivRet$(\sigma_{out^*}, VK_{x^*}, RK_{x^*}) \rightarrow y^*$;
If $k$ = accept and $y^* \neq F(x^*)$, output 1; else output 0.
```

A confidentiality-preserving publicly verifiable computation scheme π is secure for a class of functions $\mathscr{F}$, if, for any F from $\mathscr{F}$ and any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\mathrm{EXP}_{\mathscr{A}}[\pi, F, \lambda] = 1\right] \leq negl(\lambda). \quad (6)$$

Here $negl(\,)$ represents a negligible function in λ.

Next we give the confidentiality definition of CP-PVC which is not defined in existing PVC frameworks [2, 16]. In this paper we focus on the confidentiality for final result, which means that the adversaries cannot learn any information about the value $y = F(x)$ from the value σ_{out} output by Compute algorithm. Here the adversaries refer to the cloud server and any nonclient verifier. Since the cloud server has extra knowledge of the evaluation key compared with the nonclient verifiers, we only need to define the result confidentiality to cloud server. And the confidentiality to cloud server implicitly implies the confidentiality to nonclient verifiers. Notice that we do not emphasize the input privacy as a necessity in publicly verifiable computation. This is because, in some scenarios, input data is obtained from some public sources that can be accessed by anyone. However, we still present a loose definition on input privacy for multivariate function, which we call $(\delta - \epsilon)$-privacy. Intuitively, it means that, for a function with an input set of multi-independent variables, the probability that the adversary leans the values of a δ fraction of the input sets is ϵ. The definitions are as follows.

Definition 5 (result confidentiality). A confidentiality-preserving publicly verifiable computation protocol π is result-confidential for a class of functions $\mathscr{F}$ if, for any F from $\mathscr{F}$, any tuple (SK_F, EK_F, PP) output by KeyGen$(1^{\lambda}, F)$, any x chosen from Domain(F), any tuple (σ_x, VK_x, RK_x) output by ProbGen(PP, SK_F, x), any σ_{out} output by Compute(EK_F, σ_x), and any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\left[\mathscr{A}\left(PP, EK_F, VK_x, \sigma_x, \sigma_{out}\right) = F(x)\right] \leq negl(\lambda). \quad (7)$$

Definition 6 ($\delta - \epsilon$ input privacy). A confidentiality-preserving publicly verifiable computation protocol π for a class of multivariate functions $\mathscr{F}$ achieves $\delta - \epsilon$ input privacy if, for any F from $\mathscr{F}$, any tuple (SK_F, EK_F, PP) output by KeyGen$(1^{\lambda}, F)$, any input set $x = \{x_i\}_{1\leq i\leq m}$ chosen from Domain(F), any tuple (σ_x, VK_x, RK_x) output by ProbGen(PP, SK_F, x), any σ_{out} output by Compute(EK_F, σ_x), and any PPT adversary $\mathscr{A}$, it holds that

$$\Pr\Big[\mathscr{A}\left(PP, EK_F, VK_x, \sigma_x, \sigma_{out}\right) = \left\{x_j\right\}_{1\leq j\leq m'} \mid x_j \in x, m' > \delta m\Big] \leq \epsilon + negl(\lambda). \quad (8)$$

Finally, we give the definition of efficiency. Informally speaking, efficiency means that the total computational cost on the client side by engaging the CP-PVC scheme is less than that of executing the direct algorithm to compute the target function. In the amortized model, since the KeyGen is done once and amortized by multiple function evaluation with different input value, this part of computational overhead does not need to be counted in.

Definition 7 (efficiency). A confidentiality-preserving publicly verifiable computation protocol π for a class of multivariate functions $\mathscr{F}$ is efficient if, for any F from $\mathscr{F}$ and any x chosen from Domain(F), the total computational cost of algorithms ProbGen and PrivRet is less than that of directly evaluating F on x.

4. The CP-PVC Scheme for Polynomial Evaluation

In this section, we first review the construction of PRF in [29], showing that it is closed form efficient for polynomials in m variables and degree at most d in each variable. Then we present the corresponding algorithm for evaluating the PRF and its closed form efficiency. After that, we give the concrete construction of our CP-PVC scheme for polynomial evaluation together with the analysis and experimental simulation.

4.1. Algorithm for PRF with Closed Form Efficiency. Let $\mathscr{G}$ be a group generator that takes as input a secure parameter λ and outputs a description of group with prime order. Consider any polynomial $p(x_1, \ldots, x_m)$ that has m variables and degree at most d in each variable. Then the polynomial has totally $l = (d+1)^m$ monomials. Index them with tuple $(i_1, \ldots, i_m)$, $0 \leq i_j \leq d$. We say that the construction of $\mathrm{PRF}_{\mathrm{LW}}(\mathbf{KG}, R_K(i_1, \ldots, i_m))$ admits the closed form efficiency for the following computation:

$$Poly\left(\{r_{(i_1,\dots,i_m)}\}_{0\le i_1,\dots,i_m\le d, x_1,\dots,x_m}\right) = \prod_{0\le i_1,\dots,i_m\le d} r_{(i_1,\dots,i_m)}^{(x_1^{i_1}\cdots x_m^{i_m})} \\ = g^{p(x_1,\dots,x_m)} \tag{9}$$

where $p(\cdot)$ is the polynomial whose coefficients are the discrete logs of the r values. If we set $r_{(i_1,\dots,i_m)} = R_K(i_1,\dots,i_m)$, then there exists an algorithm $\mathbf{CFEval}_{\mathbf{Poly},z}(K, x_1,\dots,x_m)$ that can compute

$$g^{p(x_1,\dots,x_m)} = \prod_{0\le i_1,\dots,i_m\le d} R_K(i_1,\dots,i_m)^{(x_1^{i_1}\cdots x_m^{i_m})} \tag{10}$$

in time $O(m\log d)$, instead of the regular running time $O(d^m\cdot m\log d)$.

The proof of the above claim can be found in [2]. Here we show the algorithm for evaluating the PRF as well as the polynomial $P(x_1,\dots,x_m)$.

Let $s = \lfloor \log d\rfloor + 1$. The construction of PRF is the following algorithm:

(i) $\mathbf{KG}(1^\lambda, m, s)$: Run $\mathcal{G}(1^\lambda)$ to generate a group description $(p, g, \mathbb{G}_1)$. Choose $4ms+2$ random values

$$y_0, z_0, \{y_{a,b}, z_{a,b}, w_{a,b}, v_{a,b}\}_{1\le a\le m, 1\le b\le s} \xleftarrow{\$} \mathbb{Z}_{\mathbb{p}}. \tag{11}$$

The algorithm outputs

$$K = \left(y_0, z_0, \{y_{a,b}, z_{a,b}, w_{a,b}, v_{a,b}\}_{a,b}\right). \tag{12}$$

The domain of the function is $\{0,\dots,d\}^m$, and the range is $\mathbb{G}_1$.

(ii) $R_K(\cdot)$: Let $i = (i_1,\dots,i_m)$ be the input of the PRF. First interpret each $i_j = (i_{j,1},\dots,i_{j,s})$ as a binary string of s bits. Then run Algorithm 1.

Finally, the value of the PRF is $R_K(i) = g^{R_K^1(i)}$. Let $x = (x_1,\dots,x_m)$ be the input; the polynomial p can be written as

$$p(x_1,\dots,x_m) = \sum_{0\le i_1,\dots,i_m\le d} R_K^1(i_1,\dots,i_m)\, x_1^{i_1}\cdots x_m^{i_m}. \tag{13}$$

Now we present the algorithm for evaluating the computation $g^{p(x_1,\dots,x_m)}$ in Algorithm 2. The derivation and proof can be found in [2].

Finally, the value of $g^{p(x_1,\dots,x_m)} = g^{p^1(x)}$. Note that the above algorithm makes totally ms times recursive operation. Thus its running time is $O(ms) = O(m\log d)$, much faster than the regular running time $O(d^m\cdot m\log d)$.

With the use of the above closed form efficient PRF in Algorithm 2, we can realize public verifiability by letting the PRF value be a part of the verification key. Then our remaining goal is to make this public verification process "blind". Intuitively, to make the value $F(x)$ confidential, one way is to encrypt the input value x. This is usually done in verifiable computation schemes. However such operation requires the homomorphic property of the encryption scheme, and encryption schemes with better homomorphic

```
set $R_K^1(i) = y_0$, $R_K^2(i) = z_0$
for $j = 1$ to $m$: do
  for $k = 1$ to $s$: do
    if $i_{j,k} = 0$ then
      $R_K^1(i) = R_K^1(i)$, $R_K^2(i) = R_K^2(i)$
    else
      $R_K^1(i) = R_K^1(i)\cdot y_{j,k} + R_K^2(i)\cdot z_{j,k}$
      $R_K^2(i) = R_K^1(i)\cdot w_{j,k} + R_K^2(i)\cdot v_{j,k}$
    end if
  end for
end for
return $g^{R_K^1(i)}$
```

Algorithm 1: PRF.

```
set $p^1(x) = y_0$, $p^2(x) = z_0$
for $j = 1$ to $m$: do
  for $k = 1$ to $s$: do
    $p^1(x) = p^1(x) + x_j^{2^{k-1}}(p^1(x)\cdot y_{j,k} + p^2(x)\cdot z_{j,k})$
    $p^2(x) = p^2(x) + x_j^{2^{k-1}}(p^1(x)\cdot w_{j,k} + p^2(x)\cdot v_{j,k})$
  end for
end for
return $g^{p^1(x)}$
```

Algorithm 2: $\mathbf{CFEval}_{\mathbf{Poly}}(K, x_1,\dots,x_m)$.

property are usually less efficient. Therefore, here we consider the way to blind the target polynomial F. We use two sparse m-variate polynomials $A(x)$ and $B(x)$ to randomize the target polynomial. Here sparse means that the degree of the polynomial is 1 and there are at most m terms of monomials that are nonzero. Thus the value of $F(x)$ is covered under $A(x)$ and $B(x)$, and the computational overhead of $A(x)$ and $B(x)$ is $O(m)$.

4.2. Construction of CP-PVC for Polynomial Case.

4.2. Construction of CP-PVC for Polynomial Case. Now, we present our concrete scheme of CP-PVC for multivariate polynomials F with m variables and degree at most d' of each variable. In the case, F has totally at most $w = (d'+1)^m$ terms of monomial. Assume $T(x) = A(x)F(x)+B(x)$. Let the degree of each variable be at most d and the total terms of monomial be $l = (d+1)^m$, and for $1\le j_T\le l$ write $j_T = (i_1^T,\dots,i_m^T)$ with $0\le i_k^T\le d$; then we can represent $T(x)$ by $\sum_{j_T=1}^{l}\hat{t}_{j_T}\cdot t_{j_T}(x)$, where $t_{j_T}(x)$ represents each monomial and $\hat{t}_{j_T}$ represents the coefficient correspondingly. Similarly, represent $A(x)$ and $B(x)$ by $\sum_{j_A=1}^{2^m}\hat{a}_{j_A}\cdot a_{j_A}(x)$ and $\sum_{j_B=1}^{2^m}\hat{b}_{j_B}\cdot b_{j_B}(x)$, respectively. The CP-PVC scheme works as follows:

KeyGen: For parameter λ, the client runs a bilinear group generator $\mathcal{G}_e(1^\lambda)$ to generate a bilinear tuple $(p, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$. Choose a tuple $\{\hat{a}_{j_A}, \hat{b}_{j_B}\}_{1\le j_A, j_B\le 2^m} \xleftarrow{\$} \mathbb{G}_1$.

The client runs PRF key generation algorithm $\mathbf{KG}(1^\lambda, \lfloor \log d \rfloor + 1, m)$ to generate a key K and the range in $\mathbb{G}_1$. Choose a random $\alpha \in \mathbb{Z}_p$, and compute

$$E_{j_T} = g^{\alpha \cdot \hat{t}_{j_T}} \cdot R_K(j_T), \quad \forall j_T = 1, \dots, l. \tag{14}$$

Let $E = (E_1, \dots, E_l) \in \mathbb{G}_1^l$. The algorithm outputs

$$\begin{aligned} SK_F &= \left(K, \left\{\hat{a}_{j_A}, \hat{b}_{j_B}\right\}_{1 \le j_A, j_B \le 2^m}\right), \\ EK_F &= (T, E), \\ PP &= \left(e(g,h)^\alpha, e\right). \end{aligned} \tag{15}$$

ProbGen: On inputting $SK_F = (K, \{\hat{a}_{j_A}, \hat{b}_{j_B}\}_{1 \le j_A, j_B \le 2^m})$, $PP = (e(g,h)^\alpha, e)$, and owning x, the client computes $A(x)$,$B(x)$ and $\mathbf{CFEval_{Poly}}(K, x_1, \dots, x_m) = g^{R(x)}$. The algorithm outputs

$$\begin{aligned} \sigma_x &= x, \\ VK_x &= e\left(g^{R(x)}, h\right), \\ RK_x &= (A(x), B(x)). \end{aligned} \tag{16}$$

Compute: On receiving $EK_F = (T, E)$ and $\sigma_x = x$, the cloud server computes $y' = T(x) = \sum_{j_T=1}^{l} \hat{t}_{j_T} \cdot t_{j_T}(x)$, $V = \prod_{j_T=1}^{l} E_{j_T}{}^{t_{j_T}(x)}$ and outputs

$$\sigma_{out} = \left(y', V\right). \tag{17}$$

PubVer: On receiving $VK_x = e(g^R(x), h)$ and $\sigma_{out} = (y', V)$, the verifier checks the equation

$$e(V, h) = \left(e(g,h)^\alpha\right)^{y'} \cdot VK_x. \tag{18}$$

The algorithm outputs *accept* if the above equation holds; otherwise it outputs *reject*.

PrivRet: On receiving $VK_x = e(g^R(x), h)$, $\sigma_{out} = (y, V)$, and owning $RK_x = (A(x), B(x))$, the algorithm first runs the PubVer algorithm with VK_x and σ_{out}. If the PubVer outputs *accept*, then compute $y = (y' - B(x))/A(x)$ and return y as the output.

4.3. Security Analysis.

Now we analyze the correctness, security, and confidentiality of the proposed CP-PVC scheme.

4.3.1. Correctness.

Correctness is easy to prove. Let σ_{out}, VK_x, RK_x be as above; then

$$\begin{aligned} e(V, h) &= e\left(\prod_{j_T=1}^{l} E_{j_T}{}^{t_{j_T}(x)}, h\right) \\ &= e\left(\prod_{j_T=1}^{l} \left(g^{\alpha \cdot \hat{t}_{j_T}} \cdot R_K(j_T)\right)^{t_{j_T}(x)}, h\right) \\ &= e\left(g^{\alpha T(x)}, h\right) \cdot e\left(g^{R(x)}, h\right) \\ &= \left(e(g,h)^\alpha\right)^{y'} \cdot VK_x. \end{aligned} \tag{19}$$

The above proves the correctness for evaluating $y' = T(x)$, and this implies the correctness of $y = (y' - B(x))/A(x) = F(x)$.

4.3.2. Confidentiality.

We show that, for an adversarial cloud server with the information of $(PP, EK_F, VK_x, \sigma_x, \sigma_{out})$, the probability that it can extract the value of the result y is negligible; i.e., the result confidentiality holds for cloud server. This implies that the confidentiality also holds for any third party verifier since they have less information (EK_F) than the cloud server does. To extract $y = F(x)$, one way is directly from $T(x)$. However, this requires the knowledge of $(A(x), B(x))$ which are parts of the retrieval key and kept secret. The other way is to discover the coefficients of the function $F(\cdot)$ and evaluate F on x since the input value x is a plain text stored in σ_x. We will show that under this circumstance the adversary cannot discover the coefficients of F with a nonnegligible probability. Let A, B, F be as follows:

$$\begin{aligned} A(x) &= \sum_{j_A=1}^{2^m} \hat{a}_{j_A} \cdot a_{j_A}(x) \\ B(x) &= \sum_{j_B=1}^{2^m} \hat{b}_{j_B} \cdot b_{j_B}(x) \\ F(x) &= \sum_{j_F=1}^{(d'+1)^m} \hat{f}_{j_F} \cdot f_{j_F}(x) \end{aligned} \tag{20}$$

and then we have that

$$\begin{aligned} T(x) &= A(x)F(x) + B(x) \\ &= \sum_{j_T=1}^{2^m} \left(\sum_{j_T = j_A + j_F} \hat{a}_{j_A} \hat{f}_{j_F} + \hat{b}_{j_B}\right) t_{j_T}(x) \\ &\quad + \sum_{j_T=2^m+1}^{l} \left(\sum_{j_T = j_A + j_F} \hat{a}_{j_A} \hat{f}_{j_F}\right) t_{j_T}(x) \\ &= \sum_{j_T=1}^{l} \hat{t}_{j_T} t_{j_T}(x). \end{aligned} \tag{21}$$

Comparing the coefficients and setting $2^m = s$ of the above equation, we get a system

$$\begin{gathered}\sum_{j_A+j_F=1} \hat{a}_{j_A}\hat{f}_{j_F} + \hat{b}_1 = \hat{t}_1 \\ \vdots \\ \sum_{j_A+j_F=s} \hat{a}_{j_A}\hat{f}_{j_F} + \hat{b}_s = \hat{t}_s \\ \sum_{j_A+j_F=s+1} \hat{a}_{j_A}\hat{f}_{j_F} = \hat{t}_s + 1 \\ \vdots \\ \sum_{j_A+j_F=l} \hat{a}_{j_A}\hat{f}_{j_F} = \hat{t}_l. \end{gathered} \tag{22}$$

Note that the above system does not have a unique solution in $\mathbb{G}_1$, and the coefficients of $A(\cdot)$ and $B(\cdot)$ are chosen uniformly at random from $\mathbb{G}_1$. This means that the probability to choose the correct coefficients is negligible, and thus the privacy of coefficients of $F(\cdot)$ is guaranteed, which makes the confidentiality of the final result hold.

4.3.3. Security. The security of the scheme is based on co-CDH assumption and the corresponding proof follows from that in [2]. We take it as an inspiration and define the following four games.

Game 0. It is the same as Experiment $\mathrm{EXP}_{\mathscr{A}}[\pi, F, \lambda]$.

Game 1. It is similar to Game 0, with the difference that EK_F contains coefficients randomly chosen from $\mathbb{G}_1$ instead of T.

Game 2. It is similar to Game 1, with the difference that the ProbGen phase uses an inefficient algorithm $\prod_{j_T=1}^{l} R_K(j_T)^{t_{j_T}(x)}$ instead of the $\mathbf{CFEval_{Poly}}(K, x_1, \ldots, x_m)$ to evaluate VK_x.

Game 3. It is similar to Game 2, with the difference that each value $R_K(j_T)$ is replaced by a random element $R_{j_T} \in \mathbb{G}_1$.

We use a hybrid way to perform the proof, with the following claims.

Corollary 4. $\Pr[G_0(\mathscr{A}) = 1] = \Pr[G_1(\mathscr{A}) = 1]$.

Proof. This claim holds in an obvious way, since the change of algorithm for evaluating VK_X does not change the distribution of its values. Thus the probability that the adversary wins is the same in both Games 0 and 1. □

Corollary 5. $|\Pr[G_1(\mathscr{A}) = 1] = \Pr[G_2(\mathscr{A}) = 1]|$.

Proof. The difference between Games 2 and 1 is the coefficients of the target function. According to the confidentiality proof, these coefficients are indistinguishable since A and B are chosen uniformly at random, thus sharing the same distribution in the view of the adversary. □

Corollary 6. $|\Pr[G_2(\mathscr{A}) = 1] = \Pr[G_3(\mathscr{A}) = 1]| \le \epsilon_{prf}$, *where ϵ_{prf} represents the probability in the pseudorandomness definition of PRF.*

Proof. The difference between Games 3 and 2 is that the output of PRF $R_K(x)$ is replaced by uniformly random elements in $\mathscr{G}_1$. According to the pseudorandomness property of PRF, the probability that an adversary $\mathscr{A}$ distinguishes the two values is no better than ϵ_{prf}. □

Corollary 7. $|\Pr[G_3(\mathscr{A}) = 1]| \le \epsilon_{cdh}$, *where ϵ_{cdh} represents the probability that an adversary solves co-CDH problem.*

Proof. To prove this claim we need to show that if there exists a PPT algorithm that wins in Game 3 with a probability larger than ϵ_{cdh}, then one can build an efficient algorithm $\mathscr{B}$ with oracle access to $\mathscr{A}$ to solve the co-CDH problem with some nonnegligible probability. Assume that the group in co-CDH problem is described as $(p, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ and the adversary is given a co-CDH tuple (g^a, h^b) with the exponents a, b chosen randomly from $\mathbb{Z}_p$; the algorithm $\mathscr{B}$ works as follows.

First, $\mathscr{B}$ needs to simulate public parameter PP and evaluation key EK_F. It computes the bilinear map $e(g^a, h^b)$ and chooses $2l$ random elements $\{\hat{t}_{j_T}, E_{j_T}\}_{1\le j_T\le l}$. The simulated public key $PP = (e(g^a, h^b), e)$, and the simulated evaluation key $EK_F = (T, E) = \{\hat{t}_{j_T}, W_{j_T}\}_{1\le j_T\le l}$. Since the W generated in Game 3 contains a PRF as a factor and the PRF enjoys pseudorandomness, it is clear that the PP and EK_F generated above enjoy a perfect distribution as that in Game 3.

Second, $\mathscr{B}$ gives the simulated PP and EK_F to $\mathscr{A}$ and prepares to simulate the answers to the queries from $\mathscr{A}$. A difference here is that $\mathscr{B}$ does not need to generate the retrieval key RK in querying phase as that in Game 3 because $\mathscr{A}$ does not have access to RK, neither does $\mathscr{B}$ need this RK to solve the co-CDH problem. Let x_i $(1 \le i \le q)$ be the querying value. $\mathscr{B}$ first computes $U_{j_T} = e(E_{j_T}, h)/e(g^a, h^b)^{\hat{t}_{j_T}}$ and then computes $VK_{x_i} = \prod_{j_T=1}^{l} U_{j_T}{}^{t_{j_T}(x_i)}$. Since in Game 3,

$$\frac{e\left(E_{j_T}, h\right)}{e\left(g^{\alpha}, h\right)^{\hat{t}_{j_T}}} = R_K\left(j_T\right), \tag{23}$$

combining the bilinear property of map e, it is clear that the simulated VK_{x_i} also enjoys a perfect distribution as that in Game 3. $\mathscr{A}$ is then given the corresponding verification key VK_{x_i}. Let x^* be the challenge value chosen by $\mathscr{A}$. The above process is repeated and $\mathscr{A}$ obtains VK_{x^*}.

Finally, $\mathscr{A}$ will output a tuple $\hat{\sigma}_{out^*} = (\hat{y}, \widehat{V})$ such that

$$\begin{gathered}\mathrm{PubVer}\left(VK_{x^*}, \sigma_{out^*}\right) \longrightarrow k, \\ \mathrm{PrivRet}\left(\sigma_{out^*}, VK_{x^*}, RK_{x^*}\right) \longrightarrow y^*,\end{gathered} \tag{24}$$

where $k = accept$ and $y^* \ne F(x^*)$. Due to the correctness property, this $y^* \ne F(x^*)$ also implies that $\hat{y} \ne T(x^*)$. By $k = accept$ we obtain that

$$e\left(\widehat{V}, h\right) = e\left(g^a, h^b\right)^{\hat{y}} \cdot VK_{x^*}. \tag{25}$$

Let $y = F(x^*)$ be the correct output of the PrivRet algorithm. Then by correctness we know $y' = T(x^*)$ is also correct. Thus the following equation also holds:

$$e(V, h) = e\left(g^a, h^b\right)^{y'} \cdot VK_{x^*}, \tag{26}$$

where $V = \prod_{j_T=1}^{l} E_{j_T}{}^{t_{j_T}(x^*)}$.

Dividing (25) and (26) and combining bilinear property, we can obtain that

$$e\left(\frac{\widehat{V}}{V}, h\right) = e\left(g^{ab}, h\right)^{\widehat{y}-y'}. \tag{27}$$

Then $\mathscr{B}$ can compute $g^{ab} = (\widehat{V}/V)^{1/(\widehat{y}-y')}$. Thus, if $\mathscr{A}$ wins Game 3 with probability ϵ_{cdh}, $\mathscr{B}$ solves the co-CDH problem with the same probability. This proves Claim 4. □

Combining the four claims together, we obtain that

$$\Pr\left[\mathrm{EXP}_{\mathscr{A}}\left[\pi, F, \lambda\right] = 1\right] \leq \epsilon_{prf} + \epsilon_{cdh}. \tag{28}$$

4.4. Performance Analysis. In this subsection, we analyze the computational complexity of the proposed scheme and compare it with some existing works [2, 19]. Our scheme is efficient in the amortized model. That is, the expensive KeyGen algorithm is executed once and its computational overhead is amortized by the following evaluation of the polynomial function with different input value. Here the structure of the function cannot be changed unless another KeyGen algorithm for a different function is performed. Now we give detailed analysis on the computational costs of each algorithm. Let each letter denote the same item as that in the proposed scheme above. In the KeyGen phase, to compute the term E_{j_T}, the client needs to execute $2(d' + 2)^m + 1$ times of exponential modulo and $(d' + 1)^m$ times of multiplication arithmetic in group $\mathbb{G}_1$. To compute the structure of function $T(\cdot)$, $(2(d'+1))^m$ times of multiplication and 2^m times of addition arithmetic in $\mathbb{N}$ are executed. To generate the public key, one time of pairing arithmetic is executed. In the ProbGen phase, no execution needs to be done to generate σ_x since σ_x remains the same as input x. To generate the public verification key VK_x, algorithm $\mathbf{CFEval_{Poly}}(\cdot)$ is executed together with one time pairing arithmetic. And to generate the private retrieval key, the client needs to execute two times of (m-variate, 1-degree) sparse polynomial evaluation arithmetic. In the Compute phase, the cloud server needs to execute one time of (m-variate,d' + 1-degree) polynomial evaluation arithmetic to compute y' and $(d' + 2)^m$ times of exponential modulo in group $\mathbb{G}_1$ to compute term V. In the PubVer phase, to compute the corresponding terms in the checking equation, the third party verifier needs to execute one time of pairing arithmetic, one time of exponential modulo arithmetic in $\mathbb{G}_T$, and one time of multiplication arithmetic in $\mathbb{G}_T$, respectively. Finally, in the PrivRet phase, the client firstly needs to execute what is done in PubVer phase and then execute, respectively, one time of subtraction and division arithmetic in $\mathbb{N}$ to compute the result y.

We present a complexity comparison between our scheme and existing works of PVC schemes for polynomial evaluation [2, 19]. After analyzing the specific arithmetic of each algorithm, We would like to use the notation O to compare computational complexity. Since the arithmetic operations in different groups are the same, we use three types of notation, O_I,O_N, and O_E, to classify different types of complexity. O_I denotes the time complexity of operations in ring $\mathbb{Z}_p$ including integer addition and multiplication modulo prime p. O_N denotes the time complexity of operations in ring $\mathbb{N}$ including addition and multiplication. O_E denotes the time complexity of operations in group $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ including multiplication, exponentiation, and bilinear map. The comparison result is shown in Table 1. From the table we can see that, compared with the scheme in [2], ours has a complexity increase of $O_N(d'^m)$ in KeyGen, $O_N(m)$ in ProbGen, and $O_N(1)$ in PrivRet performed by the client. However, since the most expensive arithmetic in each of the three phases is $O_E(d'^m)$, $O_I(m \log d')$, and $O_E(1)$, respectively, the impact of the increasing complexity will not be obvious in the real performance. Compared with the scheme in [19], we first notice that the scheme in [19] only works for univariate case, while ours works for multivariate case. If we set "m = 1", the complexity of the two schemes is nearly the same, with a small increase of $O_N(1)$ in ProbGen and PrivRet respectively, which is not the most expensive part in the two phases. By doing so, our scheme achieves a security property-result confidentiality that is considered to be important for user privacy.

We also provide some experimental simulation of CP-PVC for polynomial to show the efficiency. Intuitively, efficiency means that the total computational overhead on the client side by engaging an outsourcing computation protocol is less than that by directly executing the target function algorithm. Since our scheme is constructed in an amortized model, i.e., the computations in KeyGen need to be done only once, we will not count that part of computation in the efficiency performance. Then for the client side, all the computation costs that are counted contain ProbGen and PrivRet phase. We implement the corresponding algorithms on the client side using MATLAB 2015 on a computer with Intel(R) Core(TM) i7-4790 CPU processor running at 3.60 GHz and 8 GB RAM. We show the time cost for different sizes of problems in Table 2, where m represents variable number and s represents bit length of the highest degree d'. This part of work is done in the previous conference version of this paper [24].

For comparison of efficiency, we also do some evaluation using the direct algorithm $F(x) = \sum_{j_F=1}^{w} \widehat{f}_{j_F} \cdot f_{j_F}(x)$, where $f_{j_F}(x)$ represents each monomial for polynomial F and $w = (2^s + 1)^m$. The corresponding result is shown in Table 3, which is also done in the previous conference version [24]. We see that the cost time is already infinite when the case goes to s = 50. This is reasonable because theoretically the complexity of the direct algorithm is $O(m \cdot 2^s)$. Recalling that the complexity of the way using closed form efficient PRF is $O(ms)$, thus the scheme achieves a log time efficiency promoted for the evaluation of multivariate polynomials. We

Table 1: Comparison of the three PVC schemes for polynomial evaluation.

	FG12 [2]	EOA16 [19]	Ours
Problem size	m-variate, d'-degree	univariate, d'-degree	m-variate, d'-degree
KeyGen	$O_E(d'^m)$	$O_E(d') + O_N(d')$	$O_E(d'^m) + O_N(d'^m)$
ProbGen	$O_E(1) + O_I(m \log d')$	$O_E(1) + O_I(1)$	$O_E(1) + O_I(m \log d') + O_N(m)$
Compute	$O_E(d'^m) + O_N(d'^m)$	$O_E(d') + O_N(d')$	$O_E(d'^m) + O_N(d'^m)$
PubVer	$O_E(1)$	$O_E(1)$	$O_E(1)$
PrivRet	$\perp$	$\perp$	$O_E(1) + O_N(1)$
Result confidentiality	No	No	Yes

Table 2: Times cost of the client [24].

m	$s = 10$	$s = 50$	$s = 200$	$s = 1000$
1000	0.0312s	0.0936s	0.1248s	0.4836s
5000	0.0624s	0.1716s	0.4836s	2.1372s
20000	0.1872s	0.5304s	1.7628s	8.3304s
100000	0.5460s	2.3712s	8.5956s	41.7614s

Table 3: Times cost of direct algorithm [24].

m	$s = 10$	$s = 50$	$s = 200$	$s = 1000$
1000	3.28s	∞	∞	∞
5000	6.55s	∞	∞	∞
20000	19.65s	∞	∞	∞
100000	59.24s	∞	∞	∞

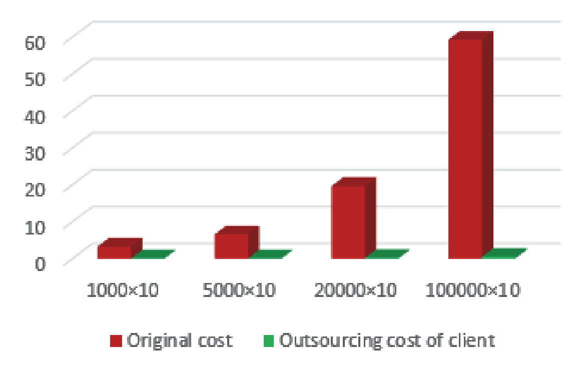

Figure 1: Time cost of different problem sizes [24].

show in Figure 1 a time cost comparison between direct algorithm and outsourcing algorithm, in which case $s = 10$; i.e., the degree of each variate is 2^{10}. This is also done in the previous conference version [24].

5. The CP-PVC Scheme for Matrix-Vector Multiplication

In this section, we first review the PRF that is used for matrix-vector case and give the algorithm for its closed form efficiency. Then we give the concrete construction of our CP-PVC scheme for matrix-vector multiplication together with the analysis and experimental simulation.

5.1. Algorithm for PRF with Closed Form Efficiency. The PRF is defined in another domain, namely, the set $[1, \ldots, m] \times [1, \ldots, n]$.

Let $\mathcal{G}$ be a group generator that takes as input a secure parameter λ and outputs a description of group with prime order. The PRF is defined as follows:

(i) $\mathbf{KG}(1^\lambda, m, n)$: Run $\mathcal{G}(1^\lambda)$ to generate a group description $(p, g, \mathbb{G}_1)$. Choose $2(n+m)$ random values

$$\{A_i, B_i\}_{1 \le i \le m} \overset{\$}{\leftarrow} \mathbb{G}_1, \qquad \{\alpha_j, \beta_j\}_{1 \le j \le n} \overset{\$}{\leftarrow} \mathbb{Z}_p. \tag{29}$$

The algorithm outputs

$$K = \{A_i, B_i, \alpha_j, \beta_j\}_{i,j}. \tag{30}$$

(ii) $R_K(\cdot)$: Let (i, j) be the input of the PRF. The algorithm outputs

$$R_K(i, j) = A_j^{\alpha_i} B_j^{\beta_i}. \tag{31}$$

The pseudorandomness holds under Decisional Linear assumption, and the corresponding proof is shown in [29]. We omit it here.

Now we consider the n-dimensional vector $\vec{t} = (t_1, \ldots, t_n) \in \mathbb{G}$, where $t_i = \prod_j r_{i,j}{}^{x_j}$ and $(r_{i,j})$ forms an $n \times m$ matrix. We show that the construction of $\mathrm{PRF}_M(\mathbf{KG}, R_K(i, j))$ admits the closed form efficiency for the computation of vector $\vec{t}$. If we set $r_{i,j} = R_K(i, j)$, then there exists an algorithm $\mathbf{CFEval_{Matrix}}(K, x_1, \ldots, x_m)$ that can compute $\vec{t} = (t_1, \ldots, t_m)$ in time $O(m+n)$, instead of the regular running time $O(nm)$. The corresponding algorithm is shown in Algorithm 3.

With the use of the above closed form efficient PRF in Algorithm 3, we can realize public verifiability by letting the PRF value be a part of the verification key. Then our remaining goal is to make this public verification process "blind". Inspired by the blinding technique in Chen et al.'s work [15], we can use a vector $\vec{r}$ to blind the real input $\vec{r}$ and a matrix $N \in \mathbb{Z}_p^{n \times m}$ to blind the target matrix M and ask the cloud server to compute $(M+N)(\vec{x} + \vec{r})$. Thus the

```
set $\overrightarrow{t} = \Phi$
$A = \prod_{j=1}^{m} A_j^{x_j}$
$B = \prod_{j=1}^{m} B_j^{x_j}$
**for** $i = 1$ to $n$: **do**
    $t_i = A^{\alpha_i} B^{\beta_i}$
**end for**
**return** $\overrightarrow{t} = (t_1, \ldots, t_n)$
```

ALGORITHM 3: $\mathbf{CFEval}_{\mathbf{Matx}}(K, x_1, \ldots, x_m)$.

result is confidential and the public verification phase can be processed as usual. However, to retrieve the real result $M\overrightarrow{x}$, the user needs to compute the value $M\overrightarrow{r}N\overrightarrow{x}$ and $N\overrightarrow{r}$. This will cost three times the original computational task $M\overrightarrow{x}$ does ($O(nm)$). To reduce the overhead of this computation, we can make the blinding vector and matrix sparse. The sparse matrix is like that in [15]. To make the result confidentiality solid, we usually set at least one nonzero element in each row of matrix N. For the sparse vector, we set randomly $\lceil (1-\delta)m \rceil$ $(0 < \delta < 1)$ positions 0. Then the computational overhead of $M\overrightarrow{r}$, $N\overrightarrow{x}$, and $N\overrightarrow{r}$ is reduced to $O(\delta nm)$, $O(n)$, and $O(\delta n)$. What is more, the blinding vector also preserves the privacy of input data to some extent.

5.2. Construction of CP-PVC for Matrix-Vector Case. Now, we present our concrete scheme of CP-PVC for matrix multiplication. Let p be a prime. Recall that our goal is to compute the n-dimensional vector $\overrightarrow{y} = M\overrightarrow{x}$, where M is an $n \times m$ matrix denoted by $(M_{i,j})$ with entries from $\mathbb{Z}_p$. Such matrix-vector multiplication can be naturally extended to matrix-matrix multiplication by regarding the latter matrix as a row vector of column vectors and repeating the above matrix-vector multiplication several times. For matrix-vector multiplication $M\overrightarrow{x}$, the CP-PVC scheme works as follows:

(i) KeyGen: Let $M = (M_{i,j})$. For parameter λ, the client runs a bilinear group generator $\mathcal{G}_e(1^\lambda)$ to generate a bilinear tuple $(p, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \widehat{e})$. Choose a random sparse matrix $N \in \mathbb{Z}_p^{n\times m}$ and set $M' = M + N$. The client then runs PRF key generation algorithm $\mathbf{KG}(1^\lambda, m, n)$ to generate a key K and the range in $\mathbb{G}_1$. Choose a random $\alpha \in \mathbb{Z}_p$, and compute

$$E_{i,j} = g^{\alpha \cdot M'_{i,j}} \cdot R_K(i,j), \quad \forall i = 1, \ldots, n;\ j = 1, \ldots, m. \quad (32)$$

Let $E = (E_{i,j}) \in \mathbb{G}_1^{n\times m}$. The algorithm outputs

$$SK_F = (K, N), \\ EK_F = (E, M'), \\ PP = (\widehat{e}, \widehat{e}(g,h)^\alpha). \quad (33)$$

(ii) ProbGen: On inputting $SK_F = (K, N)$, $PP = (\widehat{e}, \widehat{e}(g,h)^\alpha)$, and owning $\overrightarrow{x} = (x_1, \ldots, x_m) \in \mathbb{Z}_p^m$, the client randomly chooses a $\delta \in (0,1)$ and a sparse m-dimensional vector $\overrightarrow{r} = (r_1, \ldots, r_m) \in \mathbb{Z}_p^m$, where $\lceil (1-\delta)m \rceil$ coordinates of $\overrightarrow{r}$ are 0. Let $\overrightarrow{x}' = \overrightarrow{x} + \overrightarrow{r}$. Compute $M\overrightarrow{r}$, $N\overrightarrow{x}$, $N\overrightarrow{r}$ in $O(\delta nm)$, $O(n)$, $O(\delta n)$ time, respectively, and compute $t_i = \mathbf{CFEval}_{\mathbf{Matx}}(K, x'_1, \ldots, x'_m)$ in $O(n+m)$ time using the closed form efficiency. Define $v_i = \widehat{e}(t_i, h)$ for $i = 1, \ldots, n$.

The algorithm outputs

$$\sigma_x = \overrightarrow{x}', \\ VK_x = (v_i, \ldots, v_n), \\ RK_x = (M\overrightarrow{r}, N\overrightarrow{x}, N\overrightarrow{r}). \quad (34)$$

(iii) Compute: On receiving $EK_F = (E, M')$ and $\sigma_x = \overrightarrow{x}'$, the cloud server computes $\overrightarrow{y}' = M'\overrightarrow{x}'$, $w_i = \prod_{j=1}^{m} E_{i,j}^{x'_j}$ (for $i = 1, \ldots, n$) and outputs

$$\sigma_{out} = (\overrightarrow{y}', W) = ((y'_1, \ldots, y'_n), (w_i, \ldots, w_n)). \quad (35)$$

(iv) PubVer: On receiving $VK_x = (v_i, \ldots, v_n)$ and $\sigma_{out} = (\overrightarrow{y}', W)$, the verifier checks the following equation:

$$\widehat{e}(w_i, h) = \widehat{e}(g, h)^{\alpha y'_i} \cdot v_i, \quad \forall i = 1, \ldots, n. \quad (36)$$

The algorithm outputs *accept* if the above equation holds; otherwise, it outputs *reject*.

(v) PrivRet: On receiving $VK_x = (v_i, \ldots, v_n)$, $\sigma_{out} = (\overrightarrow{y}', W)$, and owning $RK_x = (M\overrightarrow{r}, N\overrightarrow{x}, N\overrightarrow{r})$, the client first runs the PubVer algorithm with VK_x and σ_{out}. If the PubVer outputs *accept*, then compute $\overrightarrow{y} = \overrightarrow{y}' - M\overrightarrow{r} - N\overrightarrow{x} - N\overrightarrow{r}$ and return $\overrightarrow{y}$ as the output.

Remark 8. In order to be uniform, we assume that each entry of the matrix M and vector $\overrightarrow{x}$ is from ring $\mathbb{Z}_p$. However, when performing the arithmetic computation between matrix-matrix, vector-vector, and matrix-vector, the execution is actually done in the ring $\mathbb{N}$. That is, the results are not in the form modulo prime p. This is to keep correctness of the computational result.

5.3. Security Analysis. Now we analyze the correctness, security, and privacy and confidentiality of the proposed CP-PVC scheme.

5.3.1. Correctness. Correctness is easy to prove. Let σ_{out}, VK_x, RK_x be as above. For $i = 1, \ldots, n$,

$$\begin{aligned}\hat{e}(w_i,h) &= \hat{e}\left(\prod_{j=1}^{m} E_{i,j}^{x_j'}, h\right)\\ &= \hat{e}\left(\prod_{j=1}^{m}\left(g^{\alpha\cdot M_{i,j}'}\cdot R_K(i,j)\right)^{x_j'}, h\right)\\ &= \hat{e}\left(\prod_{j=1}^{m} g^{\alpha M_{i,j}' x_j'}, h\right)\\ &\quad\cdot \hat{e}\left(\prod_{j=1}^{m}(R_K(i,j))^{x_j'}, h\right)\\ &= \hat{e}\left(g^{\alpha y_i'}, h\right)\cdot\hat{e}(t_i,h) = \hat{e}(g,h)^{\alpha y_i'}\cdot v_i.\end{aligned} \tag{37}$$

The above proves the correctness for evaluating $\overrightarrow{y}' = M\overrightarrow{x}'$, and this implies the correctness of $\overrightarrow{y} = \overrightarrow{y}' - M\overrightarrow{r} - N\overrightarrow{x} - N\overrightarrow{r}$.

5.3.2. Result Confidentiality and Input Privacy. The result confidentiality is similar to the polynomial case. We prove that, given the tuple $(PP, EK_F, VK, \sigma_x, \sigma_{out})$, the adversary cannot extract the result vector $y = Mx$.

Let $M, N, \overrightarrow{x}, \overrightarrow{r}, \overrightarrow{y}, \overrightarrow{y}'$ be as above; then we have that

$$y_i = \sum_{j=1}^{m} M_{i,j}\cdot x_j \tag{38}$$

and

$$\begin{aligned}y_i' &= \sum_{j=1}^{m}\left(M_{i,j}+N_{i,j}\right)\cdot\left(x_j+r_j\right)\\ &= \sum_{j=1}^{m} M_{i,j}\cdot r_j + \sum_{j=1}^{m} N_{i,j}\cdot x_j + \sum_{j=1}^{m} N_{i,j}\cdot r_j + \sum_{j=1}^{m} M_{i,j}\\ &\quad\cdot x_j.\end{aligned} \tag{39}$$

Notice that the blinding matrix has at least one nonzero element in each row, and the nonzero elements are all chosen uniformly at random. Thus the y_i are all blinded by a random nonzero element, which makes the confidentiality of $\overrightarrow{y}$ hold.

As an additional bonus, the random sparse vector $\overrightarrow{r}$ that is added to the input $\overrightarrow{x}$ provides some preservation to the privacy of input value. Since the vector is sparse, with $\lceil(1-\delta)m\rceil$ coordinates being 0, such blinding operation may preserve the input privacy at a degree of $1-\delta$; namely, the adversary may successfully guess one entry of the input vector with this probability. However since the position of 0 elements is chosen at random, the probability of the adversary to recover $1-\delta$ fraction of the input data is $1/\binom{m}{\delta m}$.

5.3.3. Security. The proof of the security follows that in [2]. We take it as an inspiration and define the following four games.

Game 0. It is the same as Experiment $\mathrm{EXP}_{\mathscr{A}}[\pi, F, \lambda]$.

Game 1. It is similar to Game 0, with the difference that EK_F contains an $n\times m$ random matrix from $\mathbb{Z}_p^{n\times m}$ instead of N.

Game 2. It is similar to Game 1, with the difference that the ProbGen phase uses an inefficient algorithm $\prod_{j=1}^{m} R_K(i,j)^{x_j}$ instead of the $\mathbf{CFEval_{Poly}}(K, x_1, \ldots, x_m)$ to evaluate VK_x.

Game 3. It is similar to Game 2, with the difference that each value $R_K(i,j)$ is replaced by a random element $R_{i,j}\in\mathbb{G}_1$.

We use a hybrid way to perform the proof, with the following claims.

Corollary 4. $\Pr[G_0(\mathscr{A})=1] = \Pr[G_1(\mathscr{A})=1]$.

Proof. This claim holds in an obvious way, since the change of algorithm for evaluating VK_X does not change the distribution of its values. Thus the probability that the adversary wins is the same in both Games 0 and 1. □

Corollary 5. $|\Pr[G_1(\mathscr{A})=1] = \Pr[G_2(\mathscr{A})=1]|$.

Proof. The difference between Games 2 and 1 is the coefficients of the target matrix. According to the confidentiality proof, these coefficients are indistinguishable since each $N_{i,j}$ is chosen uniformly at random, thus sharing the same distribution in the view of the adversary. □

Corollary 6. *$|Pr[G_2(\mathscr{A})=1] = Pr[G_3(\mathscr{A})=1]| \le \epsilon_{prf}$, where ϵ_{prf} represents the probability in the pseudorandomness definition of PRF.*

Proof. The difference between Games 3 and 2 is that the output of PRF $R_K(x)$ is replaced by uniformly random elements in $\mathbb{G}_1$. According to the pseudorandomness property of PRF, the probability that an adversary $\mathscr{A}$ distinguishes the two values is no better than ϵ_{prf}. □

Corollary 7. *$|Pr[G_3(\mathscr{A})=1]| \le \epsilon_{cdh}$, where ϵ_{cdh} represents the probability that an adversary solves co-CDH problem.*

The proof of this corollary is the same as that in [2]; we omit it here.

Combining the four claims together, we obtain that

$$\Pr\left[\mathrm{EXP}_{\mathscr{A}}[\pi, F, \lambda] = 1\right] \le \epsilon_{prf} + \epsilon_{cdh}. \tag{40}$$

5.4. Performance Analysis. We also analyze the computational complexity of the proposed CP-PVC scheme for matrix-vector multiplication and compare it with existing works [2, 19]. Analogous to polynomial case, we first give detailed analysis on the computational costs of each algorithm in matrix-vector multiplication case. Let each letter denote the same item as that in the proposed scheme above. In the KeyGen phase, to compute the term E_{j_T}, the client needs to execute $3nm+1$ times of exponential modulo and $2nm$ times of multiplication arithmetic in group $\mathbb{G}_1$. To compute

Table 4: Comparison of the three PVC schemes for matrix vector multiplication.

	FG12 [2]	EOA16 [19]	Ours
KeyGen	$O_E(nm)$	$O_E(nm) + O_N(nm)$	$O_E(nm) + O_N(n)$
ProbGen	$O_E(n+m)$	$O_E(m)$	$O_E(n+m) + O_N(\delta nm)$
Compute	$O_E(nm) + O_N(nm)$	$O_E(nm) + O_N(nm)$	$O_E(nm) + O_N(nm)$
PubVer	$O_E(n)$	$O_E(n)$	$O_E(n)$
PrivRet	$\perp$	$\perp$	$O_E(n) + O_N(n)$
Result confidentiality	No	No	Yes
Input Privacy	No	No	Partial

Table 5: Time cost under different δ values [24].

$n \times m$	Original cost	VK_Gen	RK_Gen ($\delta = 0.5$)	RK_Gen ($\delta = 0.2$)
1000×50000	1.4508s	0.0077s	0.6864s	0.2652s
1000×100000	3.3228s	0.0156s	1.2728s	0.5460s
1000×200000	7.9092s	0.0312s	2.9484s	1.0764s
1000×400000	14.1683s	0.0624s	6.4428s	2.1840s

the matrix M', since N is sparse, only n times of addition arithmetic in $\mathbb{N}$ are executed. To generate the public key, one time of pairing arithmetic is executed. In the ProbGen phase, the client first needs to execute δm times of addition arithmetic in $\mathbb{N}$ to get σ_x. To generate the public verification key VK_x, algorithm $\mathbf{CFEval_{Matx}}(\cdot)$ is executed together with n times of pairing arithmetic. And to generate the private retrieval key, the client needs to execute three times of matrix-vector multiplication arithmetic with computation complexity being $O(\delta nm)$, $O(n)$, and $O(\delta n)$, respectively. In the Compute phase, the cloud server needs to execute one time of dense matrix-vector multiplication arithmetic to compute y' and nm times of exponential modulo and $n(m-1)$ times of multiplication in group $\mathbb{G}_1$ to compute term W. In the PubVer phase, to compute the corresponding terms in the checking equation, the third party verifier needs to execute n times of pairing arithmetic, n times of exponential modulo arithmetic in $\mathbb{G}_T$, and n times of multiplication arithmetic in $\mathbb{G}_T$, respectively. Finally, in the PrivRet phase, the client firstly needs to execute what is done in PubVer phase and then execute, respectively, three times of vector subtraction arithmetic in $\mathbb{N}$ to compute the result y.

Next we present a complexity comparison between our scheme and existing works of PVC schemes for matrix-vector multiplication [2, 19]. Similar to the polynomial case, We also use the notation O to compare computational complexity. According to the specific analysis of each algorithm, the arithmetic computations take place in ring $\mathbb{N}$ and group $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$. Thus we only use two types of notation, O_N and O_E, where O_N denotes the time complexity of operations in ring $\mathbb{N}$ including addition and multiplication and O_E denotes the time complexity of operations in group $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ including multiplication, exponentiation, and bilinear map. Let $n \times m$ be the problem size; then the comparison result is shown in Table 4. From the table we can see that, compared with the scheme in [2], ours has a complexity increase of $O_N(\delta nm)$ in ProbGen and $O_N(n)$ in PrivRet performed by the client. However, when the value of δ is small, the part $O_N(\delta nm)$ will be much less expensive than $O_E(n+m)$. And $O_N(n)$ is surely less expensive than $O_E(n)$. Therefore, by choosing a proper δ, the impact of the increasing complexity will not be obvious in the real performance. Compared with the scheme in [19], ours has a complexity increase of $O_E(n) + O_N(\delta nm)$ in ProbGen and $O_N(n)$ in PrivRet performed by the client. If $n \leq m$, or the difference between n and m is small, the complexity of $O_E(m)$ and $O_E(n+m)$ is nearly the same. Under this circumstance, our scheme can achieve a comparative performance with a small δ chosen. What is more, our scheme achieves result confidentiality and partial input privacy that are considered to be important for user privacy.

We also provide an experimental evaluation on the client side to show the efficiency compared with direct algorithm for matrix-vector multiplication. Different from that in polynomial case, we have a sparse coefficient δ in matrix-vector case. We choose different δ values and show the time cost for different sizes of problems in Table 5, which is done in the previous conference version of this paper [24]. Note that in the ProbGen phase, the time cost of VK generation is related only to n, m but not to δ. However, the time cost of RK generation is related to not only n, m but also δ. With a smaller δ, one can get a more efficient CP-PVC scheme, but this inevitably brings more risks to input privacy. This seems inherently a trade-off. Also, note that the computations in the PubVer and PrivRet phase are n times of bilinear map operation and three times of vector subtraction, respectively. This part of cost is too small to be counted in, so we omit it from the table.

From the table we can see that, no matter what the choice of δ is, the total cost of VK generation and RK generation is much less than the original cost of matrix multiplication. Things can be more visualized in the column diagram of Figures 2 and 3 [24], where the volume of VK generation (blue) and RK generation (green) together is less than that of the original cost (red). Even when counting in the omitted time cost of PubVer and PrivRet phase, the CP-PVC scheme achieves an acceptable efficiency performance in practice.

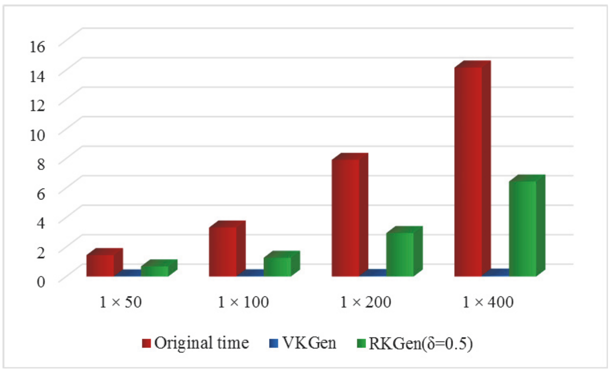

FIGURE 2: Time cost of different problem sizes ($\times 1000$) when $\delta = 0.5$ [24].

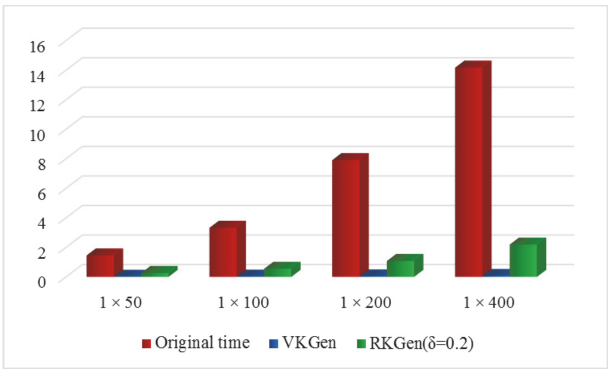

FIGURE 3: Time cost of different problem sizes ($\times 1000$) when $\delta = 0.2$ [24].

6. Conclusion

We have proposed confidentiality-preserving publicly verifiable computation schemes for multivariate polynomial evaluation and matrix-vector multiplication. The proposed schemes achieve both public verifiability and result confidentiality, while the latter one also achieves partial input privacy. These security properties are considered to be important in practical application of publicly verifiable computation. With the experimental evaluation we have done, the scheme is also acceptable in practice.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Nature Science Foundation of China under Grants No. 61272091 and 61772311.

References

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[2] D. Fiore and R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS '12)*, pp. 501–512, ACM, October 2012.

[3] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers[M]," in *Advances in Cryptology C CRYPTO*, vol. 6223 of *Lecture Notes in Comput. Sci.*, pp. 465–482, Springer, Berlin, 2010.

[4] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164, 1982.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of Computing (STOC '09)*, pp. 169–178, ACM, Bethesda, Md, USA, 2009.

[6] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, pp. 97–105, 2010.

[7] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proceedings of the 21st ACM Conference on Computer and Communications Security, CCS 2014*, pp. 844–855, usa, November 2014.

[8] J. Lai, R. H. Deng, H. Pang, and J. Weng, "Verifiable Computation on Outsourced Encrypted Data," in *Computer Security - ESORICS 2014*, vol. 8712 of *Lecture Notes in Computer Science*, pp. 273–291, Springer International Publishing, Cham, 2014.

[9] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets[C]," in *Proceedings of the Conference on Advances in Cryptology*, Lecture Notes in Comput. Sci., pp. 111–131, Springer, Heidelberg.

[10] M. Backes, D. Fiore, and M. Reischuk R, "Verifiable delegation of computation on outsourced data[C]," in *Proceedings of ACM Sigsac Conference on Computer & Communications Security*, pp. 826–828, 2013.

[11] L. F. Zhang and R. Safavi-Naini, "Batch verifiable computation of outsourced functions," *Designs, Codes and Cryptography. An International Journal*, vol. 77, no. 2-3, pp. 563–585, 2015.

[12] P. Ananth, N. Chandran, V. Goyal, B. Kanukurthi, and R. Ostrovsky, "Achieving privacy in verifiable computation with multiple serversCwithout FHE and without pre-processing[C]," in *International Workshop on Public Key Cryptography*, vol. 8383 of *Lecture Notes in Comput. Sci.*, pp. 149–166, Springer, Heidelberg, 2014.

[13] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communication Security, ASIACCS 2010*, pp. 48–59, chn, April 2010.

[14] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1172–1181, 2013.

[15] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of large-scale systems of linear equations," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 69–78, 2015.

[16] B. Parno, M. Raykova, and V. Vaikuntanathan, "How to Delegate and Verify in Public: Verifiable Computation from Attribute-Based Encryption[C]," in *Proceedings of the International Conference on Theory of Cryptography*, pp. 422–439, 2012.

[17] Y. Sun, Y. Yu, X. Li, K. Zhang, H. Qian, and Y. Zhou, "Batch verifiable computation with public verifiability for outsourcing

polynomials and matrix computations," in *Proceedings of the Part I of Information Security and Privacy - 21st Australasian Conference, ACISP 2016*, vol. 9722, pp. 293–309, Springer International Publishing, Melbourne, VIC, Australia, 2016.

[18] C. Papamanthou, E. Shi, and R. Tamassia, "Signatures of Correct Computation," in *Theory of Cryptography*, vol. 7785 of *Lecture Notes in Computer Science*, pp. 222–242, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[19] K. Elkhiyaoui, M. Önen, M. Azraoui, and R. Molva, "Efficient techniques for publicly verifiable delegation of computation," in *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2016*, pp. 119–128, chn, June 2016.

[20] J. Alderman, C. Cid, and J. Crampton, "Publicly Verifiable Outsourced Computation with a Key Distribution Centre[J]," Eprint Arxiv, 2014:1406.

[21] J. Alderman, C. Janson, C. Cid, and J. Crampton, "Revocation in publicly verifiable outsourced computation," in *Information security and cryptology*, vol. 8957 of *Lecture Notes in Comput. Sci.*, pp. 51–71, Springer, Cham, 2015.

[22] J. Alderman, C. Janson, C. Cid, and J. Crampton, "Hybrid Publicly Verifiable Computation[M]," in *Topics in Cryptology - CT-RSA*, pp. 147–163, Springer International Publishing, 2016.

[23] J. Ye, H. Zhang, and C. Fu, "Verifiable delegation of polynomials," *International Journal of Network Security*, vol. 18, no. 2, pp. 283–290, 2016.

[24] J. Sun, B. Zhu, J. Qin, J. Hu, and Q. Wu, "Confidentiality-preserving publicly verifiable computation," *International Journal of Foundations of Computer Science*, vol. 28, no. 6, pp. 799–818, 2017.

[25] D. Boneh, I. Mironov, and V. Shoup, "Secure Signature Scheme from Bilinear Maps[M]," in *Topics in Cryptology - CT-RSA*, pp. 98–110, Springer, Berlin, 2003.

[26] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *IACR cryptology ePrint archive*, 2004.

[27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing[C]," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 514–532, Springer, Berlin, 2001.

[28] P. Golle and I. Mironov, "Uncheatable Distributed Computations[C]. Topics in Cryptology - CT-RSA 2001," in *Proceedings of the The Cryptographer's Track at RSA Conference 2001*, vol. 2001, pp. 425–440, San Francisco, CA, USA.

[29] A. B. Lewko and B. Waters, "Efficient pseudorandom functions from the decisional linear assumption and weaker variants," in *Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09*, pp. 112–120, usa, November 2009.