An Efficient and Provably Secure Private Polynomial Evaluation Scheme

Zhe Xia[1], Bo Yang[2(✉)], Mingwu Zhang[3], and Yi Mu[4]

[1] School of Computer Science, Wuhan University of Technology, Wuhan 430070, China
xiazhe@whut.edu.cn

[2] School of Computer Science, Shaanxi Normal University, Xi'an 710062, China
byang@snnu.edu.cn

[3] School of Computers, Hubei University of Technology, Wuhan 430068, China
csmwzhang@gmail.com

[4] School of Computing and Information Technology, University of Wollongong, Wollongong 2522, Australia
ymu@uow.edu.au

Abstract. Private Polynomial Evaluation (PPE) allows the service provider to outsource the computation of a polynomial to some third party (e.g. the Cloud) in a verifiable way. And meanwhile, the polynomial remains hidden to the clients who are able to query the service. In ProvSec 2017, Bultel et al. have presented the formal security definitions for PPE, including *polynomial protection* (PP), *proof unforgeability* (UNF) and *indistinguishability against chosen function attack* (IND-CFA). They have introduced a PPE scheme that satisfies all these properties, and they have also shown that a polynomial commitment scheme in Asiacrypt 2010, called $\mathsf{PolyCommit}_{\mathsf{Ped}}$, enjoys these properties as well. In this paper, we introduce another provably secure PPE scheme, which not only has computational advantages over these two existing ones, but also relies on a much weaker security assumption. Moreover, we further explore how our PPE scheme can be implemented in the distributed fashion, so that a number of third parties jointly respond to the query but none of them could learn the polynomial unless they all collude.

1 Introduction

Mathematical models have various applications in our everyday life. For example, a patient collects her medical data such as blood pressure, body temperature and heart rate by sensors, and the expert system can use some pre-defined mathematical model to evaluate her health status. A farmer collects the data of the soil, such as humidity, acidity and thermal parameters, and the agricultural consultant can use some well analysed mathematical model to predict the state

C. Su and H. Kikuchi (Eds.): ISPEC 2018, LNCS 11125, pp. 595–609, 2018.
https://doi.org/10.1007/978-3-319-99807-7_38

of the soil for the next year. The benefits for these examples are obvious: the patient gets better medical treatments, and the farmer obtains precise information regarding how much seeds to buy and when to plant them in the coming year. As the development of computer and communication technologies, things could get even better. The service provider can outsource the computation of the mathematical model to some third party, e.g. the Cloud. In this way, the service provider reduces its operation cost, because it does not need to maintain the resources of computation, storage and communication. And meanwhile, the client could access the service more conveniently, e.g. the patient can be monitored continuously in real-time.

However, the above attractive features and economical initiatives cannot succeed unless the following issues have been well addressed. On one hand, to protect its intellectual property, the service provider may not be willing to reveal the mathematical model to the clients. On the other hand, the clients may not trust the third party, and would like to verify that the mathematical model has been computed correctly. To harmonise these two contradicting requirements, several cryptographic solutions have been proposed recently in the literature. One subclass of these solutions focusing on the case where the mathematical model can be expressed as univariable polynomials are called private polynomial evaluation (PPE) schemes [6].

In a PPE scheme, as illustrated in Fig. 1, the service provider outsources the evaluation of the polynomial $f(\cdot)$ to a third party and it broadcasts some public information vk. The paid client can query the service by submitting the input data x to the third party. After evaluating the polynomial, the third party returns $f(x)$ as well as a proof π to the client. Finally, the client is able to verify whether the polynomial has been evaluated correctly using the public information vk and the proof π. Note that during this process, the client should not learn any information of the polynomial $f(\cdot)$. This not only requires that the client cannot derive the entire polynomial $f(\cdot)$, but also requires that even if the client has some prior knowledge of two polynomials $f_0(\cdot)$ and $f_1(\cdot)$, she cannot distinguish which one has been used to evaluate her input data.

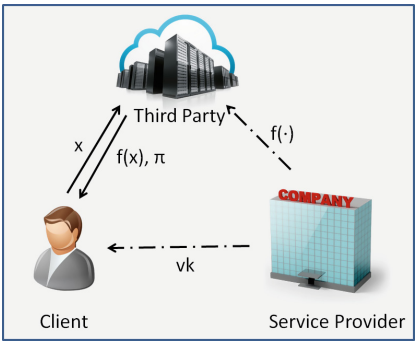

Fig. 1. An illustration of the PPE scheme

1.1 Related Works

Verifiable computation, first introduced by Gennaro et al. in [13], requires that the party who performs the computation to prove the correctness of output. Hence, it allows the service provider who has limited resources to delegate expensive computations to some untrusted parties. Furthermore, if the correctness of output can be checked by anyone who is interested, it is called publicly verifiable computation [19]. The formal security model and definitions of verifiable computation have been presented by Canetti et al. in [8]. Afterwards, this technique has been further extended in various aspects. For example, Choi et al. [9] have extended verifiable computation to the multi-client setting. Papamanthou et al. [17] have introduced the concept of signatures of correct computation, which uses multivariate polynomials for verification. Fiore and Gennaro [11] have proposed a verifiable computation scheme for polynomial evaluation and matrix computation. Parno et al. [18] have demonstrated, using a concrete prototype called Pinocchio, that some of the verifiable computations are practical in the real use. The research focus of the above works is that the verification of the proof should require less computational costs than computing the function from scratch, but they have not considered protecting the function against the client which is required in PPE.

Another related work was introduced by Naor and Pinkas [16], called oblivious polynomial evaluation (OPE), in which the service provider has some polynomial $f(\cdot)$, and the client has some input x. After the execution of the protocol, the client should obtain $f(x)$ but not the original polynomial $f(\cdot)$, and the service provider should not learn x. Although OPE and PPE shares some similarities, they still differ in several aspects: OPE does not consider verifying the correctness of polynomial evaluation, and PPE does not consider protection of x from the service provider.

Recently, several works have tried to address these two contradicting requirements simultaneously, so that the client can verify that the function has been correctly computed, and meanwhile, the function is not revealed to the client. To simplify the design, most of these works have restricted the function as univariate polynomials. In ProvSec 2017, Bultel et al. [6] called this type of schemes as private polynomial evaluation (PPE), and they presented the formal security definitions for PPE. Informally, a PPE scheme should satisfy the following three properties: (1) *polynomial protection* (PP) requires that the client cannot evaluate the polynomial by herself on any new input that she has not queried before; (2) *proof unforgeability* (UNF) requires that the third party cannot cheat the client using incorrect result; (3) *indistinguishability against chosen function attack* (IND-CFA) requires that the client cannot distinguish which polynomial has been evaluated even if she has some prior knowledge of the polynomials. In the same paper, Bultel et al. also showed that one of the polynomial commitment schemes in [15], called $\mathsf{PolyCommit}_{\mathsf{Ped}}$, satisfies these properties as well, although $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is originally designed as a verifiable secret sharing (VSS) scheme with constant size commitments. Note that a few other works [12,14] have introduced verifiable and privacy-preserving solutions for various applications and

they have claimed implicitly to achieve similar properties. But it was shown later that in these works, a malicious client can retrieve the entire polynomial in a single query. To the best of our knowledge, Bultel's scheme in [6] and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ in [15] are the existing ones that can achieve all the above three properties, and we will compare our proposed scheme with these two schemes.

1.2 Our Contributions

The contributions of this paper are summarised as follows:

- We introduce a new PPE scheme and we formally prove that it achieves PP, UNF and IND-CFA properties. Our proposed scheme not only has computational advantages over the two existing ones, but also relies on a much weaker security assumption. Regarding the computational costs, in one aspect, our scheme uses Pedersen's VSS [20] to replace Feldman's VSS [10] as used in [6], and the benefit is that we no longer need to use any CPA encryption scheme and zero-knowledge proof in order to achieve the IND-CFA property. In the other aspect, although Pedersen's VSS has been used as the main building block both in our scheme and in $\mathsf{PolyCommit}_{\mathsf{Ped}}$, the client's verification of polynomial evaluation in our scheme does not need any expensive pairing computation. Regarding the security assumptions, our scheme only relies on the discrete logarithm (DL) assumption, while Bultel's scheme needs the decisional Diffie-Hellman (DDH) assumption and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ needs the t-strong Diffie-Hellman (t-SDH) assumption [4]. It is well known that DL is a much weaker assumption than DDH and t-SDH. Moreover, the t-SDH assumption is not as well analysed by researchers in computational number theory as the other two, and the DDH assumption may fail in some special groups, e.g. the bilinear map [5].
- We further explore how our proposed scheme can be implemented in the distributed fashion. A number of third parties jointly evaluate the polynomial, but none of them can learn the secret polynomial unless they all collude. In the client's view, the polynomial appears to have been evaluated by a single third party. Note that this extension could better reflect the demands in real world applications, since the service provider may wish to keep the polynomial private from the third party as well.

1.3 Organisation of the Paper

The rest of the paper is organised as follows: In Sect. 2, we outline some preliminaries. The model and definitions of PPE are described in Sect. 3. Our proposed PPE scheme as well as its security proofs are presented in Sect. 4. We further extend the PPE scheme into a distributed version and briefly sketch its security in Sect. 5. Finally, we conclude in Sect. 6.

2 Preliminaries

2.1 Notations

In the paper, all participants are assumed to be probabilistic polynomial time (PPT) algorithms with respect to the security parameter λ, unless stated otherwise. We use standard notions for expressing probabilistic algorithms and experiments. If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and a random coin r. We denote $y \leftarrow A(x_1, x_2, \ldots; r)$ as the experiment of picking r at random and assigning y as $A(x_1, x_2, \ldots; r)$. If S is a finite set, then $x \overset{R}{\leftarrow} S$ denotes the operation of picking an element uniformly from S. $\mathsf{Pr}[x \leftarrow S; y \leftarrow T; \ldots : p(x, y, \ldots)]$ is denoted as the probability that the predicate $p(x, y, \ldots)$ will be true after the ordered execution of the algorithms $x \leftarrow S, y \leftarrow T$, etc. A function $\epsilon(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^+$ is called negligible if for all $c > 0$, there exists a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. Moreover, let p, q be large primes such that $q|p-1$, and G is a subgroup of $\mathbb{Z}_p^*$ with order q. Both g and h are generators of G, but it is required that nobody knows $\log_g h$. We assume that all computations are modulo p unless stated otherwise.

2.2 Building Blocks

Pedersen's Verifiable Secret Shairng [20]**.** Secret sharing is a useful technique to ensure secrecy and availability of sensitive information. The dealer can share the secret among a number of participants, so that a quorum of these participants work together can recover the secret, but less participants cannot learn any information of the secret. However, in traditional secret sharing schemes, the dealer may cheat by distributing inconsistent shares, and the participants may cheat by revealing fake shares when reconstructing the secret. Using verifiable secret sharing (VSS), these dishonest behaviours can be detected. Pedersen's VSS is based on Shamir secret sharing, and it works as follows:

- The dealer first generates two polynomials $f(\cdot)$ and $f'(\cdot)$ over $\mathbb{Z}_q$ with degree k as:

$$f(z) = a_0 + a_1 z + \ldots + a_k z^k \qquad f'(z) = b_0 + b_1 z + \ldots + b_k z^k$$

  where the secret $s = a_0$.
- Then dealer publishes the commitments $C_i = g^{a_i} h^{b_i}$ for $i = 0, 1, \ldots, k$.
- The x_i values for $i = 1, 2, \ldots, n$ are public parameters associate with each participant such that $x_i \neq x_j$ if $i \neq j$. For each participant, the dealer computes the share $s_i = f(x_i)$ and $s_i' = f'(x_i)$.
- Once receiving the share s_i and s_i', each participant verifies its validity by:

$$g^{s_i} h^{s_i'} = \prod_{j=0}^{k} (C_i)^{{x_i}^j}$$

If the above verification fails, a participant can make an accusation against the dealer. Note that the same verification also can be used to prevent the participants from revealing fake shares when reconstructing the secret.

Homomorphic Secret Sharing [3]**:** Denote $(s_1, s_2, \ldots, s_n)$ as a set of shares encoding the secret s, and $(s'_1, s'_2, \ldots, s'_n)$ as another set of shares encoding the secret s'. Moreover, $\oplus$ and $\otimes$ are denoted as the operation of shares and the operation of the secret, respectively. Secret sharing is said to have the homomorphic property if the set $(s_1 \oplus s'_1, s_2 \oplus s'_2, \ldots s_n \oplus s'_n)$ encodes the secret $s \otimes s'$. It is obvious that Pedersen's VSS enjoys the $(+,+)$-homomorphic property, where the symbol $+$ denotes the addition operation in the group $\mathbb{Z}_q$. We will use this property to extend our PPE scheme into the distributed version.

3 Model and Definitions

3.1 Private Polynomial Evaluation (PPE)

A PPE scheme [6] is specified by the following four randomised algorithms: Setup, Init, Compute, Verif:

- Setup: takes as input the security parameter λ, and returns the system parameters params.
- Init: takes as input params, and returns some public information vk associated with the secret polynomial $f(\cdot) \in \mathbb{Z}_q[X]$.
- Compute: takes as inputs params, vk, the polynomial $f(\cdot)$ and the client's input x, and returns $y = f(x)$ as well as some proof π.
- Verif: takes as inputs params, vk, x, y and π, and returns 1 if accepting the evaluation of $f(\cdot)$ on x, and returns 0 otherwise.

3.2 Security Properties and Assumptions

Definition 1 (k-polynomial protection (k-PP)): *A PPE scheme is said to be k-PP secure if there exists a negligible function $\epsilon(\cdot)$ such that for all PPT adversaries $\mathcal{A}_{PP}$, we have:*

$$\mathsf{Pr}\Big[\mathsf{params} \leftarrow \mathsf{Setup}(\lambda); f(\cdot) \leftarrow \mathbb{Z}_q[X]_k; \mathsf{vk} \leftarrow \mathsf{Init}(\mathsf{params}, f(\cdot)); \Sigma \leftarrow \emptyset; (x^*, y^*) \leftarrow {\mathcal{A}_{PP}}^{\mathcal{O}_{PP}(\cdot)}(\mathsf{params}, \mathsf{vk}): f(x^*) = y^* \wedge (x^*, y^*) \notin \Sigma\Big] < \epsilon(\lambda)$$

In the above expression, $f(\cdot)$ is a polynomial over $\mathbb{Z}_q$ with degree k, and $\mathcal{O}_{PP}$ is an oracle that takes as input x and returns $f(x)$ as well as some proof π. The adversary $\mathcal{A}_{PP}$ is restricted to query $\mathcal{O}_{PP}$ at most k times. The set Σ records all pairs $(x, f(x))$ that have been queried.

Definition 2 (Proof unforgeability (UNF)): *A PPE scheme is said to be UNF secure if there exists a negligible function $\epsilon(\cdot)$ such that for all PPT adversaries $\mathcal{A}_{UNF}$, we have:*

$$\mathsf{Pr}\Big[\mathsf{params} \leftarrow \mathsf{Setup}(\lambda); f(\cdot) \leftarrow \mathcal{A}_{UNF}(\mathbb{Z}_q[X]_k); \mathsf{vk} \leftarrow \mathsf{Init}(\mathsf{params}, f(\cdot)); (x^*, y^*, \pi^*) \leftarrow \mathcal{A}_{UNF}(\mathsf{params}, \mathsf{vk}, f(\cdot)): f(x^*) \neq y^* \wedge \mathsf{Verif}(\mathsf{params}, \mathsf{vk}, x^*, y^*, \pi^*) = 1\Big] < \epsilon(\lambda)$$

Definition 3 (Indistinguishability against chosen function attack (IND-CFA)): *A PPE scheme is said to be IND-CFA secure if there exists a negligible function $\epsilon(\cdot)$ such that for all PPT adversaries $\mathcal{A}_{CFA}$, we have:*

$$\mathsf{Pr}\Big[\mathsf{params} \leftarrow \mathsf{Setup}(\lambda); (f_0(\cdot), f_1(\cdot)) \leftarrow \mathcal{A}_{CFA}(\mathbb{Z}_q[X]_k); b \overset{R}{\leftarrow} \{0,1\}; \mathsf{vk} \leftarrow \mathsf{Init}(\mathsf{params}, f_b(\cdot)), b^* \leftarrow {\mathcal{A}_{CFA}}^{\mathcal{O}_{CFA}(\cdot)}(\mathsf{params}, \mathsf{vk}): b^* = b\Big] < 1/2 + \epsilon(\lambda)$$

In the above expression, both $f_0(\cdot)$ and $f_1(\cdot)$ are polynomials over $\mathbb{Z}_q$ with degree k. Moreover, $f_0(\cdot)$ and $f_1(\cdot)$ agree at most k points (x_i, y_i) for $i = 1, 2, \ldots, k$. When the adversary $\mathcal{A}_{CFA}$ queries the oracle $\mathcal{O}_{CFA}(\cdot)$ using some of these x_i values, it will output the corresponding y_i as well as a proof π. Otherwise, the oracle $\mathcal{O}_{CFA}(\cdot)$ outputs the symbol $\perp$.

Definition 4 (Discrete logarithm (DL) assumption): *Given the description of the group G and $x \overset{R}{\leftarrow} \mathbb{Z}_q$, the discrete logarithm assumption implies that there exits a negligible function $\epsilon(\cdot)$ such that for all PPT adversaries $\mathcal{A}_{DL}$, we have* $\mathsf{Pr}[x^* \leftarrow \mathcal{A}_{DL}(g, g^x) : x^* = x] < \epsilon(\lambda)$.

4 Our Proposed PPE Scheme

4.1 Our Scheme

Our proposed PPE scheme contains four algorithms Setup, Init, Compute, Verif, and it works as follows:

– Setup: Given a security parameter λ, this algorithm first generates two primes p and q such that $q|p-1$. It then generates the group G which is a subgroup of $\mathbb{Z}_p^*$ with order q, and two generators g, h of G such that $\log_g h$ is unknown[1]. Finally, all these parameters are made public as params.

[1] Note that such a value h can be generated by a distributed coin flipping protocol that outputs a random value $r \in \mathbb{Z}_p^*$, followed by computing $h = r^{(p-1)/q}$ satisfying that $h \neq 1$.

- Init: Before outsourcing the polynomial $f(z) = a_0 + a_1 z + \ldots + a_k z^k$ over $\mathbb{Z}_q$ with degree k to the third party, the service provider randomly selects another polynomial $f'(z) = b_0 + b_1 z + \ldots + b_k z^k$ over $\mathbb{Z}_q$ with degree k, and computes the commitments $C_i = g^{a_i} h^{b_i}$ for $i = 0, 1, \ldots, k$. Then, the service provider sends both these polynomials $f(z)$ and $f'(z)$ to the third party using a private channel, and broadcasts the commitments as the public information vk.
- Compute: Once receiving the client's input x, the third party computes two values $y = f(x)$ and $y' = f'(x)$, and then sends back y and y' to the client. Note that in our proposed PPE scheme, the proof π is an empty string.
- Verif: The client checks the correctness of polynomial evaluation by verifying the following equation:

$$g^y h^{y'} = \prod_{i=0}^{k} (C_i)^{x^i}$$

4.2 Security Analysis

We first show that the proposed PPE scheme is correct. In other words, if the third party has correctly evaluated the polynomial $f(\cdot)$ on x, then the client's verification will always be successful. Considering the case that the third party is honest, then we have $y = f(x) = \sum_{i=0}^{k} a_i x^i$ and $y' = f'(x) = \sum_{i=0}^{k} b_i x^i$. Therefore, the following equation will always hold:

$$g^y h^{y'} = g^{\sum_{i=0}^{k} a_i x^i} h^{\sum_{i=0}^{k} b_i x^i} = \prod_{i=0}^{k} (g^{a_i} h^{b_i})^{x^i} = \prod_{i=0}^{k} (C_i)^{x^i}$$

Theorem 1. *The proposed PPE scheme achieves the k-PP property.*

Proof. Because the adversary $\mathcal{A}_{PP}$ can query the oracle $\mathcal{O}_{PP}$ at most k times, $\mathcal{A}_{PP}$ can obtains at most k points $(x_i, f(x_i))$ for $i = 1, 2, \ldots, k$. Without loss of generality, we assume that $\mathcal{A}_{PP}$ aims to compute $f(x_{x+1})$ for some value x_{x+1} that she has not queried, and we prove that $\mathcal{A}_{PP}$ can succeed only with negligible probability.

Since the secret polynomial $f(\cdot)$ is with degree k, there are $k+1$ unknown coefficients. Obtaining k points $(x_i, f(x_i))$ for $i = 1, 2, \ldots, k$ only gives k equations, hence none of these coefficients can be derived. In particular, the constant coefficient $a_0 = f(0)$ is uniformly distributed within $\mathbb{Z}_q$. Using the Lagrange interpolation, the value $f(x_{k+1})$ can be expressed as:

$$f(x_{k+1}) = f(0) \prod_{j=1}^{k} \frac{x_{k+1}}{x_j} + \sum_{i=1}^{k} f(x_i) \prod_{j=1, j\neq i}^{k} \frac{x_{k+1} - x_i}{x_j - x_i}$$

Because both $\prod_{j=1}^{k} \frac{x_{k+1}}{x_j}$ and $\sum_{i=1}^{k} f(x_i) \prod_{j=1, j\neq i}^{k} \frac{x_{k+1}-x_i}{x_j - x_i}$ are constant values, we denote them as a and b respectively. Then, the above equation can be rewritten as $f(x_{k+1}) = f(0)a + b$, which is an affine cipher. Moreover, because q is

a prime and all these values $x_i \in \mathbb{Z}_q$, we have $\gcd(a, q) = 1$. This further implies that the value $f(x_{k+1})$ will be randomly distributed within $\mathbb{Z}_q$. Therefore, the probability that $\mathcal{A}_{PP}$ correctly computes the value $f(x_{x+1})$ is exactly $1/q$, which is negligible with respect to the security parameter λ.

Theorem 2. *The proposed scheme achieves the* UNF *property under the discrete logarithm assumption.*

Proof. Suppose there exists an adversary $\mathcal{A}_{UNF}$ who violates the UNF property with non-negligible probability, then we demonstrate that $\mathcal{A}_{UNF}$ can be used to construct an algorithm that computes $\log_g h$.

Based on the definition of UNF property, the adversary $\mathcal{A}_{UNF}$ has the knowledge of the two polynomials $f(\cdot)$ and $f'(\cdot)$. Her purpose is to output a triple (x^*, y^*, y'^*) such that within the following two inequalities $y^* \neq f(x^*)$ and $y'^* \neq f'(x^*)$, at least one of them is true, and meanwhile the verification of the equation $g^{y^*} h^{y'^*} = \prod_{i=0}^{k} (C_i)^{x^{*i}}$ is satisfied.

Firstly, we prove by contradiction that if $y^* \neq f(x^*)$, then we also have $y'^* \neq f'(x^*)$. Suppose we have $y^* \neq f(x^*)$ but $y'^* = f'(x^*)$, then $g^{y^*} h^{y'^*} = \prod_{i=0}^{k} (C_i)^{x^{*i}}$ implies that $g^{y^*} h^{y'^*} = g^{f(x^*)} h^{f'(x^*)}$, which further implies that $g^{y^*} = g^{f(x^*)}$. But this contradicts the pre-condition that $y^* \neq f(x^*)$. Hence the case $y^* \neq f(x^*)$ but $y'^* = f'(x^*)$ cannot happen. For similar reasons, the case $y'^* \neq f'(x^*)$ but $y^* = f(x^*)$ cannot happen neither. Therefore, it must be the case that both inequalities $y^* \neq f(x^*)$ and $y'^* \neq f'(x^*)$ are true.

Next, we prove that such an adversary $\mathcal{A}_{UNF}$ allows us to compute $\log_g h$. Since the verification of the equation $g^{y^*} h^{y'^*} = \prod_{i=0}^{k} (C_i)^{x^{*i}}$ is satisfied, this implies that $g^{y^*} h^{y'^*} = g^{f^*} h^{f'(x^*)}$. Hence, we have $\log_g h = \frac{y^* - f(x^*)}{f'(x^*) - y'^*}$. And because we have already proved that $f'(x^*) \neq y'^*$, the discrete logarithm $\log_g h$ can be computed with the same probability as $\mathcal{A}_{UNF}$ violates the UNF property. Therefore, based on the discrete logarithm assumption, there cannot exist an adversary $\mathcal{A}_{UNF}$ who violates the UNF property with non-negligible probability.

Theorem 3. *The proposed scheme achieves the* IND-CFA *property.*

Proof. We prove this theorem using the following strategy: suppose there are two games, Game_0 and Game_1. In Game_i for $i \in \{0, 1\}$, the polynomial $f_i(\cdot)$ is selected. We then show that the adversary $\mathcal{A}_{CFA}$'s view is perfectly indistinguishable between these two games. Hence, $\mathcal{A}_{CFA}$ will output the same b^* in both games with equal probability, and this proves that $\mathcal{A}_{CFA}$ guesses b correctly with probability exactly $1/2$.

In Game_0, $\mathcal{A}_{CFA}$ will be provided with the params in the first step. Then, in the second step, $\mathcal{A}_{CFA}$ chooses a polynomial $f_0(z) = a_{0,0} + a_{0,1} z + \ldots + a_{0,k} z^k$ over $\mathbb{Z}_q$ with degree k. In the third step, the challenger selects another random polynomial $f'_0(z) = b_{0,0} + b_{0,1} z + \ldots + b_{0,k} z^k$ over $\mathbb{Z}_q$ with degree k, and compute the commitments $C_{0,i} = g^{a_{0,i}} h^{b_{0,i}}$ for $i = 0, 1, \ldots, k$. Then, the challenger publishes vk which contains all these commitments.

In Game_1, $\mathcal{A}_{CFA}$ will be provided with exactly the same params in the first step. Then, in the second step, $\mathcal{A}_{CFA}$ chooses an independent polynomial $f_1(z) = a_{1,0} + a_{1,1}z + \ldots + a_{1,k}z^k$ over $\mathbb{Z}_q$ with degree k. In the third step, the challenger selects a random polynomial $f_1'(z) = b_{1,0} + b_{1,1}z + \ldots + b_{1,k}z^k$ over $\mathbb{Z}_q$ with degree k, and compute the commitments $C_{1,i} = g^{a_{1,i}}h^{b_{1,i}}$ for $i = 0, 1, \ldots, k$. $\mathcal{A}_{CFA}$ receives the public information vk which contains all the commitments in this step.

In the first step, $\mathcal{A}_{CFA}$'s view of the two games is exactly the same because the same params is output by the challenger. In the second step, $\mathcal{A}_{CFA}$ selects a random polynomial in both games. Hence, her view is exactly the same in this step as well. In the third step, $\mathcal{A}_{CFA}$ sees $C_{0,i} = g^{a_{0,i}}h^{b_{0,i}}$ for $i = 0, 1, \ldots, k$ in Game_0, and she sees $C_{1,i} = g^{a_{1,i}}h^{b_{1,i}}$ for $i = 0, 1, \ldots, k$ in Game_1. But all these commitments are randomly distributed in $\mathbb{Z}_p^*$. Therefore, $\mathcal{A}_{CFA}$'s view in the third step is also exactly the same. Moreover, although $\mathcal{A}_{CFA}$ can query the oracle $\mathcal{O}_{CFA}$, the oracle only responses to the query when the points lie both on $f_0(\cdot)$ and $f_1(\cdot)$. Hence, the oracle $\mathcal{O}_{CFA}$ does not give $\mathcal{A}_{CFA}$ any additional power. Therefore, $\mathcal{O}_{CFA}$ cannot distinguish these two games, and she guesses b correctly with probability exactly $1/2$.

4.3 Some Comparisons

The comparison of our proposed scheme with Bultel's PPE scheme in [6] and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ in [15] is summarised as in Table 1. The description of Bultel's PPE scheme and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ can be found in the appendix.

Table 1. Comparison of the three schemes

	params size	vk size	Verif	Assumption	Model	Trusted party
Bultel's PPE	$\mathcal{O}(1)$	$\mathcal{O}(k)$	Pairing free	DDH	RO	No
$\mathsf{PolyCommit}_{\mathsf{Ped}}$	$\mathcal{O}(k)$	$\mathcal{O}(1)$	Pairing based	t-SDH	Standard	Yes
Our PPE	$\mathcal{O}(1)$	$\mathcal{O}(k)$	Pairing free	DL	Standard	No

The main advantage of $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is that the size of vk is constant, which is much smaller than the other two schemes. However, its size of params is much larger, and this offsets the previous advantage. Recall that the client needs to know both params and vk, all these three schemes have similar communication costs. To verify the correctness of polynomial evaluation, the client in $\mathsf{PolyCommit}_{\mathsf{Ped}}$ needs to perform pairing computations, and the client in Bultel's scheme needs to verify some additional zero-knowledge proofs. Our proposed scheme has some computational advantages over these two existing PPE schemes, because it is pairing free and the client only needs to perform some standard VSS verification.

In Bultel's scheme, the $\mathsf{IND\text{-}CFA}$ property relies on the DDH assumption, and the UNF property is proved in the Random Oracle (RO) model [2]. Although the RO model is of some value, it only provides heuristic proofs. In particular, it does not rule out the possibility of breaking the scheme without finding the weakness

in the hash function [7]. Therefore, when the other parameters are equal, a proof in the standard model is still preferred. The security of $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is proved in the standard model, but it needs a trusted party to initialise the params and its UNF property relies on a less standard t-SDH assumption. Our proposed scheme also has some security advantages over these two existing PPE schemes, because it does not need any trusted party, it can be proved in the standard model, and it relies on a much weaker assumption[2].

5 A Distributed PPE Scheme

In many applications, it may require that the PPE scheme also keeps the polynomial private from the third party. In this section, we introduce a natural extension of our propose PPE scheme in order to satisfy this requirement. In this distributed PPE scheme, the secret polynomial is outsourced to a number of third parties instead of a single one. These parties jointly evaluate the polynomial for the client in a verifiable way, but none of them could learn the polynomial unless they all collude.

The distributed PPE scheme is composed of the following four algorithms Setup, Init, $\mathsf{Compute}$, Verif:

- Setup: Given a security parameter λ, two primes p and q are generated such that $q|p-1$. Then a group G is generated which is a subgroup of $\mathbb{Z}_p^*$ with order q. Moreover, g and h are denoted as two generators of G such that $\log_g h$ is unknown. Finally, all these parameters are made public as params.
- Init: Suppose the service provider wants to outsource the k degree polynomial $f(z) = a_0 + a_1 z + \ldots + a_k z^k$ over $\mathbb{Z}_q$ among t independent third parties. It first randomly selects another polynomial $f'(z) = b_0 + b_1 z + \ldots + b_k z^k$ over $\mathbb{Z}_q$ with the same degree, and then computes the commitments $C_i = g^{a_i} h^{b_i}$ for $i = 0, 1, \ldots, k$. Next, the service provider randomly generates two groups of t polynomials $f_j(z) = a_{j,0} + a_{j,1} z + \ldots + a_{j,k} z^k$ and $f'_j(z) = b_{j,0} + b_{j,1} z + \ldots + b_{j,k} z^k$ over $\mathbb{Z}_q$ with degree k, for $j = 1, 2, \ldots, t$, such that $f(z) = \sum_{j=1}^{t} f_j(z)$ and $f'(z) = \sum_{j=1}^{t} f'_j(z)$, and it computes the commitments $C_{j,i} = g^{a_{j,i}} h^{b_{j,i}}$ for $j = 1, 2, \ldots, t$ and $i = 1, 2, \ldots, k$. The service provider sends a pair of polynomials $f_j(z)$ and $f'_j(z)$ to each of the third party using a private channel. Finally, it broadcasts all the commitments generated in this step as the public information vk.
- $\mathsf{Compute}$: Once receiving the client's input x, each of the third party computes two values $y_j = f_j(x)$ and $y'_j = f'_j(x)$, and sends these two values back to the client.
- Verif: The client first checks whether $C_i = \prod_{j=1}^{t} C_{ji}$ for $i = 1, 2, \ldots, k$. This verification ensures that $f(z) = \sum_{j=1}^{t} f_j(z)$ and $f'(z) = \sum_{j=1}^{t} f'_j(z)$. Note

[2] If there exists an adversary who can break the DL assumption with non-negligible probability, then an algorithm can be designed that uses this adversary as a subroutine and breaks both DDH and t-SDH assumptions with non-negligible probability.

that such a check only needs to be performed once, even if the client may query the polynomial several times. Then, for each of the third party, the client verifies whether it has correctly evaluated its assigned polynomials by checking the following equation:

$$g^{y_j} h^{y'_j} = \prod_{i=0}^{k} (C_{j,i})^{x^i}$$

If all the above checks are satisfied, the client computes $y = f(x) = \sum_{j=1}^{t} y_j$.

Note that if batch techniques [1] were used in the Verif algorithm, the client can verify all the equations at once instead of verifying t equations individually. Here, we only briefly sketch the security of the above scheme, since it is very similar as in the standard PPE scheme. Firstly, each third party evaluates a random polynomial, hence none of them learns the secret polynomial unless they all collude. The client is allowed to query the polynomial $f(\cdot)$ at most k times, and $f(\cdot)$ is with degree k. Hence, the client cannot learn any information of the polynomial and she only has negligible probability to violate the PP property. Furthermore, unless one can break the discrete logarithm assumption, the client can use the VSS equation to detect any incorrect evaluation of the polynomial, and this implies the UNF property. Finally, because all the commitments are in the Pedersen format which is information theoretically hiding, the IND-CFA property also holds in the above scheme.

6 Conclusion

As the wide deployment of Cloud platforms, PPE is a useful primitive to delegate the evaluation of secret polynomials in a verifiable way. In this paper, we introduce a new PPE scheme that satisfies all the PP, UNF and IND-CFA properties as advocated recently. And we show that compared with the existing PPE schemes with similar properties, our scheme not only has computational advantages but also relies on a much weaker assumption. Moreover, we explore how the PPE scheme can be implemented in a distributed way so that the polynomial is also kept private from the third party. We extend our proposed PPE scheme as an example, but the same method also can be used to extend the existing PPE schemes into the distributed version.

Acknowledgement. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61572303, 61772326, 61672010, 61672398), and Natural Science Foundation of Hubei Province (Grant No. 2017CFB303, 2017CFA012). We are also grateful to the anonymous reviewers for their valuable comments on the paper.

Appendix A – $\mathsf{PolyCommit}_{\mathsf{Ped}}$

The $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme [15] contains four algorithms (Setup, Init, Compute, Verif), and it works as follows:

- Setup: This algorithm is operated by a trusted party. Given the security parameter λ, it generates two cyclic groups G and G_T with prime order p such that there exists a symmetric bilinear pairing $\hat{e} : G \times G \rightarrow G_T$. It also chooses two generators g and h of G such that $\log_g h$ is unknown. Moreover, it selects $\alpha \xleftarrow{R} \mathbb{Z}_p^*$ and sets $\mathsf{params} = (G, G_T, p, \hat{e}, g, h, (g^{\alpha}, \ldots, g^{\alpha^k}), (h^{\alpha}, \ldots, h^{\alpha^k}))$.
- Init: For the secret polynomial $f(z) = a_0 + a_1 z + \ldots + a_k z^k$, the service provider chooses a random polynomial $f'(z) = b_0 + b_1 z + \ldots + b_k z^k$ over $\mathbb{Z}_p^*$ with degree k. It computes the commitment $C = \prod_{i=0}^{k} (g^{\alpha^i})^{a_i} (h^{\alpha^i})^{b_i} = g^{f(\alpha)} h^{f'(\alpha)}$ and sets $\mathsf{vk} = C$.
- Compute: Once receiving the client's input x. The third party computes $y = f(x)$ and $y' = f'(x)$. Moreover, it computes $\phi(z) = \frac{f(z)-f(x)}{z-x} = \sum_{i=0}^{k} \delta_i z^i$ and $\phi'(z) = \frac{f'(z)-f'(x)}{z-x} = \sum_{i=0}^{k} \sigma_i z^i$. It further computes $w = \prod_{j=0}^{k} (g^{\alpha^j})^{\delta_j} (h^{\alpha^j})^{\sigma_j} = g^{\phi(\alpha)} h^{\phi'(\alpha)}$. It sets the proof as $\pi = (x, y', w)$ and returns (y, π) to the client.
- Verif: The client verifies whether $\hat{e}(C, g) = \hat{e}(w, g^{\alpha - x}) \hat{e}(g^{f(x)} h^{g'(x)}, g)$. If this equation holds, the client outputs 1, and outputs 0 otherwise.

Appendix B – Bultel's PPE Scheme

The Bultel's PPE scheme [6] also contains four algorithms (Setup, Init, Compute, Verif) as follows:

- Setup: Given the security parameter λ, the service provider generates a group G with prime order p and a generator g for the group. It chooses a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, and it sets $\mathsf{params} = (G, p, g, \mathsf{H})$. Note that the hash function is only used to generate non-interactive zero-knowledge proofs.
- Init: For the secret polynomial $f(z) = a_0 + a_1 z + \ldots + a_k z^k$, the service provider picks $\mathsf{sk} \xleftarrow{R} \mathbb{Z}_p^*$ and computes $\mathsf{pk} = g^{\mathsf{sk}}$. For $i = 0, 1, \ldots, k$, it picks $r_i \xleftarrow{R} \mathbb{Z}_p^*$ and computes $c_i = g^{r_i}$ and $d_i = \mathsf{pk}^{r_i} g^{a_i}$. Note that (c_i, d_i) is an ElGamal ciphertext encrypting the commitment g^{a_i}. Finally, it sets $\mathsf{vk} = (\{c_i, d_i\}_{0 \leq i \leq k}, \mathsf{pk})$.
- Compute: Once receiving the client's input x, the third party computes $y = f(x)$. It also computes $c = \prod_{i=0}^{k} (c_i)^{x^i} = \prod_{i=0}^{k} g^{r_i \cdot x^i} = g^{r(x)}$ and $d = \prod_{i=0}^{k} (d_i)^{x^i} = (\prod_{i=0}^{k} h^{r_i \cdot x^i}) \cdot (\prod_{i=0}^{k} g^{a_i \cdot x^i}) = h^{r(x)} g^{f(x)}$ for some polynomial $r(x) = \sum_{i=0}^{k} r_i \cdot x^i$. Moreover, it generates a non-interactive zero-knowledge proof π that (c, d) is an ElGamal ciphertext encrypting $g^{f(x)}$. Finally, it return (y, π) to the client.
- Verif: Using params and vk, the client can also compute (c, d). Then, she can verify whether π is a valid non-interactive zero-knowledge proof such that (c, d) encrypts g^y. If the verification satisfies, the client outputs 1, and outputs 0 otherwise.

References

1. Bellare, M., Garay, J.A., Rabin, T.: Batch verification with applications to cryptography and checking. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 170–191. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054320
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 62–73. ACM (1993)
3. Benaloh, J.C.: Secret sharing homomorphisms: keeping shares of a secret secret (extended abstract). In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 251–260. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_19
4. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptol. **21**(2), 149–177 (2008)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
6. Bultel, X., Das, M.L., Gajera, H., Gérault, D., Giraud, M., Lafourcade, P.: Verifiable private polynomial evaluation. In: Okamoto, T., Yu, Y., Au, M.H., Li, Y. (eds.) ProvSec 2017. LNCS, vol. 10592, pp. 487–506. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68637-0_29
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM (JACM) **51**(4), 557–594 (2004)
8. Canetti, R., Riva, B., Rothblum, G.N.: Two protocols for delegation of computation. In: Smith, A. (ed.) ICITS 2012. LNCS, vol. 7412, pp. 37–61. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32284-6_3
9. Choi, S.G., Katz, J., Kumaresan, R., Cid, C.: Multi-client non-interactive verifiable computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 499–518. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_28
10. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 1987 28th Annual Symposium on Foundations of Computer Science, pp. 427–438. IEEE (1987)
11. Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 501–512. ACM (2012)
12. Gajera, H., Naik, S., Das, M.L.: On the security of "verifiable privacy-preserving monitoring for cloud-assisted mHealth systems". In: Ray, I., Gaur, M.S., Conti, M., Sanghi, D., Kamakoti, V. (eds.) ICISS 2016. LNCS, vol. 10063, pp. 324–335. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49806-5_17
13. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_25
14. Guo, L., Fang, Y., Li, M., Li, P.: Verifiable privacy-preserving monitoring for cloud-assisted mHealth systems. In: 2015 IEEE Conference on Computer Communications, INFOCOM, pp. 1026–1034. IEEE (2015)
15. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11

16. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 245–254. ACM (1999)
17. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 222–242. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_13
18. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy, SP, pp. 238–252. IEEE (2013)
19. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_24
20. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9