PriVC: Privacy Preserving Verifiable Computation

Hardik Gajera
DA-IICT
Gandhinagar, India
hardik_gajera@daiict.ac.in

Manik Lal Das
DA-IICT
Gandhinagar, India
maniklal_das@daiict.ac.in

***Abstract*—Verifiable computation in cloud setup with a privacy-preserving feature is an interesting research problem which can find application in monitoring system such as healthcare application. For example, a public cloud service provider can facilitate computation service on patients data or symptoms based on doctors prescription (e.g., prediction function) without letting the cloud server know on anything about patient's data and the patient can verify the computed response. We present a privacy-preserving verifiable computation (PriVC) scheme in which the server can authenticate a user in a privacy-preserving manner, and compute on encrypted data of the user stored in the public cloud. The server provides the proof of computation which can be verified by the user. The PriVC preserves the privacy of the user and ensures undeniability of the service offered as well as the service consumed. The PriVC scheme uses homomorphic encryption for user data encryption and a private polynomial function for computation on encrypted data. We show that the PriVC scheme is secure under indistinguishability against chosen function attack (IND-CFA), and the proof of computation is unforgeable in the standard model.**

***Index Terms*—verification, privacy, cloud security, data encryption, monitoring system**

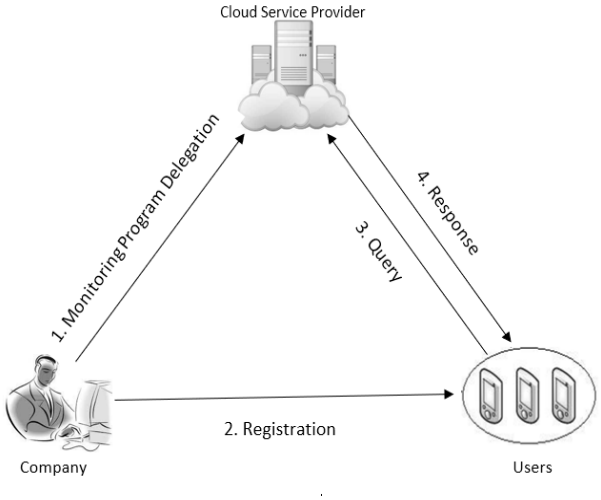

Fig. 1. A basic model for monitoring system

I. Introduction

With recent advances in machine learning, more and more predictive technologies are becoming realities. Using the training data set, a machine learning algorithm generates predictive functions, which can find applications such as healthcare, forecasting, supply-chain, etc. For example, a company (e.g., healthcare service provider) can use existing datasets of users (e.g., patients) to train a predictive function which can predict disease in the initial stage with a certain confidence. The input of the prediction function can be blood pressure, heart rate, blood glucose level, and output can be the chances of having a heart disease. A cloud-assisted monitoring system is a cost-effective computing and storage setup for convenient, on-demand data access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services. While resource outsourcing provides significant advantages to a company (data owner) as well as to customers (users), security, privacy, trust are important concerns for protecting users data from the (untrusted) cloud server and at the same time, meeting the business goal of a company. For example, the company can delegate the health monitoring systems to the cloud, where a patient can directly communicate with the cloud. However, upon receiving the patient request, the cloud can generate a fabricated report with some malicious intent. Therefore, there is a possibility that the cloud server can manipulate the data without the data owner's knowledge. To avoid such scenarios, the data owner can prefer to store data in the cloud server in an encrypted form so that the cloud server cannot manipulate the data while consumer getting services from it. A basic model of a cloud-assisted monitoring system is illustrated in Figure 1.

In recent years many proposals have been introduced in the literature on the privacy-preserving feature in the monitoring system in a cloud setup. We note that the prediction function must be a secret parameter between the company and the cloud service provider CSP (and users as well), as the company will earn financially by serving its users using the prediction function (with the help of CSP) on users data. Furthermore, the CSP may not be a trusted entity, who will instead offer this service to have financial gain by providing fast but a possibility of incorrect computation. For instance, let us look at the excerpts from Amazon Web Services(AWS) customer agreement: "*make no representations or warranties of any kind ... that the service offerings or third-party content will be uninterrupted, error-free or free of harmful components, and (IV) that any content will be secure or not otherwise lost or altered*". In such a context, the CSP is doing business and will be fair with the services, but how can one be sure about that? Therefore, to check whether the user gets services on actual data or fabricated ones, there must be a mechanism to

978-1-7281-3187-0/20/$31.00 ©2020 IEEE

prove the correctness of the computation. This requirement brings the concept of Verifiable Computation into context.

Gennaro *et al.* [2] introduced a cryptographic primitive called Verifiable Computation (VC), where an untrusted server can prove the correctness of its computation. Using verifiable computation, a computationally weak client can delegate its computation to an untrusted server. Parno *et al.* [3] gave a publicly verifiable primitive, where anyone can check the correctness of the computation. Fiore *et al.* [8] proposed a primitive for verification of polynomial evaluation and matrix multiplication. Later, Parno *et al.* [4] provided Pinocchio, a concrete system for publicly verifying arithmetic circuit. Pinocchio converts arithmetic circuits in QAP and then uses it to verify the computation of the circuit. For specific parameters, it is possible to manipulate the computation in such a way that it passes the verification test. Several other modifications of Pinocchio were proposed for specific scenarios [5]–[7]. In all these works, we observe that either the polynomial function is public or uses plain input or have flaws.

There are few schemes proposed in literature [9]–[12], [20] for verifiable computation with a hidden polynomial. Kate *et al.* [10] proposed two primitives for commitment to polynomial, where one primitive considers deterministic polynomial while other primitive considers random polynomial. Guo *et al.* [11] proposed a scheme using a bounded polynomial with the feature of privacy-preserving identity verification. Later, Gajera *et al.* [12] have shown the security weaknesses of Guo *et al.*'s scheme, that is, anyone can bypass the identity verification successfully using only public parameters, and a user can guess the secret polynomial used in the scheme [11]. Gajera *et al.* [12] also provided an improved scheme over Guo *et al.*'s scheme. Xavier *et al.* [20] proposed a formal security notion of indistinguishability of chosen function attack IND-CFA for verifiable private polynomial evaluation and showed that both the schemes [11] and [12] are not secure against IND-CFA.

A. Our Contribution

In this paper, we present a Privacy-Preserving Verifiable Computation (PriVC) for private polynomial evaluation in public cloud setup. The proposed scheme, PriVC, is the first IND-CFA secure scheme which provides a practical cloud-assisted monitoring system with the following characteristics: (i) verifiable computation on encrypted data; (ii) keeping the computation logic on data hidden from the user (secrecy of the prediction function); (iii) not letting the server know for whom the computation is intended (privacy of the user); and (iv) not denying of usage of the services by the user (undeniability).

We use homomorphic encryption for evaluating polynomial function over encrypted data which provides confidentiality of the data and allows to perform computation over encrypted data. We show that the PriVC scheme is IND-CFA secure with respect to data confidentiality, privacy-preserving user authentication, and undeniability along with verification of computation. The verification cost must be minimal in any verifiable computation scheme. The proposed scheme is implemented using SageMath version 8.1 to show that the scheme is efficient; in particular, the result verification step takes less time compared to an actual polynomial evaluation done by the CSP.

B. Organization

The rest of the paper is organized as follows: in Section 2, we give some preliminaries of cryptographic assumptions and building blocks used in our scheme. In Section 3, we provide definitions of security models related to our scheme. In Section 4, we present the PriVC scheme and its security analysis. We present the implementation details and results in section 5. We conclude our work in Section 6.

II. Preliminaries

In this section, we give the definitions of various cryptographic assumptions and functions used in our paper.

Definition 1 (Approximate GCD (A-GCD) Problem [13])**.** *Given a set of k integers of the form $x_i = q_i p + r_i$ where $q_i, p, r_i \in \mathbb{Z}$ with q_i, r_i are randomly chosen, the approximate GCD problem is to find p.*

There is no known probabilistic polynomial time algorithm in literature which can solve this problem in polynomial time.

Definition 2 (Discrete Logarithm (DL) Problem [14])**.** *Let G be a multiplicative group of order p generated by g. Given $h \in G$, the discrete logarithm (DL) problem is to find $x \in \mathbb{Z}_p^*$ such that $h = g^x$.*

There is no known probabilistic polynomial time algorithm which can solve discrete logarithm problem.

Definition 3 (Homomorphic Encryption [15])**.** *Any scheme $\varepsilon = (\mathsf{KeyGen}_\varepsilon, \mathsf{Encrypt}_\varepsilon, \mathsf{Decrypt}_\varepsilon)$ is said to be homomorphic with respect to a circuit C if and only if for any plaintexts $\pi_1, \pi_2, \ldots, \pi_n$ and ciphertexts $\psi_1, \psi_2, \ldots, \psi_n$ where $\psi_i = \mathsf{Encrypt}_\varepsilon(sk, \pi_i)$ the following equation holds true:*

$$C(\pi_1, \pi_2, \ldots, \pi_n) = \mathsf{Decrypt}_\varepsilon(sk, \mathsf{Evaluate}(C, \psi)).$$

Pisa *et al.* [16] proposed a somewhat homomorphic encryption scheme over large integers. Their scheme is an extension of DGHV scheme over bit operations [17].

Definition 4 (Extended DGHV scheme [16])**.** *The extended DGHV scheme contains following three algorithms:* KeyGen, Encryption, Decryption.

- $\mathsf{KeyGen}(\lambda)$*: The B is base parameter and the private key K_{priv} is a coprime to B in $[2^{\eta-1}, 2^\eta)$ where $\eta = \mathcal{O}(\lambda^2)$.*
- $\mathsf{Encryption}(K_{priv}, m)$*: For encrypting $m \in [0, B)$, it takes a random $r \in (-2^\rho, 2^\rho)$ and $s \in (0, 2^\gamma / K_{priv})$. The parameters $\gamma = \mathcal{O}(\lambda^5)$ and $\rho = \mathcal{O}(\lambda)$ as originally proposed in DGHV scheme [17]. It then compute ciphertext as follows:*

$$\psi = m + B \times r + s \times K_{priv}.$$

- $\mathsf{Decryption}(K_{priv}, \psi)$*: Using private key K_{priv}, it decrypts ψ as follows:*

$$m = (\psi \ mod \ K_{priv}) \ mod \ B.$$

III. Security Models

Any private polynomial evaluation scheme should be designed in such a way that the proof of computation and other public parameters must not give any advantage to an attacker. Xavier *et al.* [20] provides the formal security notion for private polynomial evaluation schemes. In our proposed PriVC scheme, we use the security notion of indistinguishability against chosen function attack (IND-CFA) as defined in Xavier *et al.*'s scheme [20]. We first define the PriVC scheme as follows.

Definition 5 (Privacy-Preserving Verifiable Computation)**.** *Let f be a polynomial in $\mathbb{Z}_q^+[X]$. A Privacy - Preserving Verifiable Computation* (PriVC) *is a 8-tuple of algorithms* (Setup, Init, KeyGen, EvalRequest, VerifyAuth, Eval, VerifyResult, VerifyTrans) *defined as:*

- $\mathsf{Setup}(\lambda)$*: It takes security parameter λ as input and the public parameters* pub*.*
- $\mathsf{Init}(f, \mathsf{pub})$*: It take polynomial function f and public parameters* pub *as input and outputs a secret key for the function,* $\mathsf{sk_f}$*, and verification key for individual user u,* $\mathsf{vk_f^u}$*.*
- $\mathsf{KeyGen}(\lambda, \mathsf{pub})$*: For a user u, it generates extended DGHV private key* $\mathsf{k_u}$ *and parameters ρ, γ.*
- $\mathsf{EvalRequest}(m, \mathsf{k_u}, \mathsf{pub}, \mathsf{vk_f^u})$*: It takes a message m, user key* k_u*, public parameters* pub*, verification key* $\mathsf{vk_f^u}$ *as input and outputs an encrypted query (c, π_a).*
- $\mathsf{VerifyAuth}(c, \pi_a, \mathsf{sk_f}, f, \mathsf{pub})$*: It takes an encrypted query (c, π_a), secret key* $\mathsf{sk_f}$*, the polynomial f and public parameters* pub *as input. It outputs c if (c, π_a) is accepted; else, aborts.*
- $\mathsf{Eval}(c, f, \mathsf{sk_f}, \mathsf{pub})$*: It takes an encrypted message c, secret key* $\mathsf{sk_f}$ *and public parameters* pub *as input. It outputs (y, π) where $y = f(c)$ is an encryption of $f(m)$ and π is* proof of computation *for $f(m)$.*
- $\mathsf{VerifyResult}(y, \pi, \mathsf{vk_f^u}, \mathsf{k_u}, m, \mathsf{pub})$*: It decrypts (y, π) and verifies the computation of $f(m)$ using verification key* $\mathsf{vk_f^u}$*.*
- $\mathsf{VerifyTrans}(c, \pi_a, \mathsf{k_u}, \mathsf{vk_f^u}, \mathsf{pub})$*: It takes a transaction $(c,\ \pi_a)$, user key* k_u*, verification key* $\mathsf{vk_f^u}$ *and public parameters* pub *as input. It outputs 1 if (c, π_a) is a valid transaction otherwise 0.*

A. Adversarial Assumptions

The polynomial $f(x)$ used in the PriVC scheme is a secret polynomial and users should not be able to learn anything about $f(x)$.

An adversary (we note that a legitimate user can also act as adversary) can choose two polynomials and then tries to guess which polynomial is used by the CSP. More concretely, given two polynomial f_0 and f_1 , the adversary should not be able to distinguish which polynomial is used by the CSP. The security notion of indistinguishability against chosen function attack (IND-CFA) is recently introduced by Xavier et al. [20]. We prove that the PriVC scheme is secure under IND-CFA model. We now give the formal definition of IND-CFA, as introduced in [20].

Definition 6 (Oracle for CFA, $\mathcal{O}_{\mathsf{CFA}}$)**.** *In this oracle, the adversary has an access to* Eval *algorithm. The adversary can query the server at most one time. The input of each query is an encrypted value c of data m and output of each query is (y, π) where $y = f_b(c) + r * (f_0(c) - f_1(c))$, r is a random integer and π is* proof *of computation. The oracle returns (y, π).*

$\mathcal{O}_{\mathsf{CFA}}\ (c, b, f_0, f_1, \mathsf{sk}_f, \mathsf{pub})$*:*
$r_1, r_2 \leftarrow \{0,1\}^*$*:*
$f' \leftarrow f_b + r_1 * (f_0 - f_1)$*:*
$\mathsf{sk}'_f \leftarrow \mathsf{sk}_f + r_2 * (f_0 - f_1)$*:*
$(y, \pi) \leftarrow \mathsf{Eval}(c, f', \mathsf{sk}'_f, \mathsf{pub})$*;*
Return (y, π).

In IND-CFA model, the adversary tries to guess which polynomial is used by a PriVC scheme. The adversary chooses two polynomials (f_0, f_1). The server randomly selects one polynomial f_b where $b \in \{0, 1\}$ and random integers r_1, r_2. It sets $f' = f_b + r_1 * (f_0 - f_1)$, $sk' = sk + r_2 * (f_0 - f_1)$ and evaluates and proves the computation of $y = f'(c)$, where c is given by the adversary. Note that if $f_0(m) = f_1(m)$, then decryption of y gives $f_b(m)$; else, the adversary gets garbage value after decryption. In the definition of IND-CFA, $\mathbb{Z}_q^+[X]_n$ represents a set of all polynomials of degree at most n with coefficients in $\mathbb{Z}_q^+$.

Definition 7 (IND-CFA [20])**.** *Let Π be a* PriVC*, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a two-party PPT adversary and n be an integer. The n-IND-CFA experiment for $\mathcal{A}$ against Π is defined as follows:*

$\mathsf{Exp}^{\textit{n-IND-CFA}}_{\Pi,\mathcal{A}}\ (\lambda)$*:*
$b \leftarrow \{0, 1\}$;
$\mathsf{pub} \leftarrow \mathsf{Setup}(\lambda)$;
$(f_0, f_1, st) \leftarrow \mathcal{A}_1(\mathsf{pub}, n)$;
$(\mathsf{sk}_f, \mathsf{vk}_f^u) \leftarrow \mathsf{Init}(f_b, \mathsf{pub})$;
$\mathsf{k}_u \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pub})$;
$(c, \pi_a) \leftarrow \mathsf{EvalRequest}(st, \mathsf{k}_u, \mathsf{pub}, \mathsf{vk}_f^u)$
$b_* \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathit{CFA}}(\cdot)}(\mathsf{k}_u, \mathsf{vk}_f^u, \mathsf{pub}, f_0, f_1, c)$;
If $f_0 \notin \mathbb{Z}_q^+[X]_n$ *or* $f_1 \notin \mathbb{Z}_q^+[X]_n$*:*
Then return $\perp$*;*
Else return $(b = b_*)$.

$\mathcal{A}$ has access to the server oracle $\mathcal{O}_{\mathsf{CFA}}(\cdot)$. The advantage of the adversary $\mathcal{A}$ against the n-IND-CFA experiment is defined by:

$$\mathsf{Adv}^{\textit{n-IND-CFA}}_{\Pi,\mathcal{A}}(\lambda) = \left| \frac{1}{2} - \Pr\left[1 \leftarrow \mathsf{Exp}^{\textit{n-IND-CFA}}_{\Pi,\mathcal{A}}(\lambda)\right] \right|.$$

A scheme Π is n-IND-CFA secure if this advantage is negligible in λ for any polynomial-time adversary $\mathcal{A}$.

We note that in the above experiment, n is the degree of the polynomial $f(x)$ and st is a testing point. Using public parameters, the adversary chooses two functions f_0, f_1 and a point st.

The second security model is regarding unforgeability property. In verifiable computation scheme, it is essential to prove the unforgeability. A PriVC scheme is said to be unforgeable when the server cannot produce valid proof for a wrong computation.

Definition 8 (Unforgeability [20])**.** *Let* Π *be a* PriVC *scheme and* $\mathcal{A} = (\mathcal{A}_1.\mathcal{A}_2)$ *be a two-party adversary. The Unforgeability (UNF) experiment for* $\mathcal{A}$ *against* Π *is defined as follows:*

$\mathsf{Exp}^{UNF}_{\Pi,\mathcal{A}}(\lambda)$:
$\mathsf{pub} \leftarrow \mathsf{Setup}(\lambda)$;
$(f, st) \leftarrow \mathcal{A}_1(\mathsf{pub})$;
$(\mathsf{sk}_f, \mathsf{vk}^u_f) \leftarrow \mathsf{Init}(f, \mathsf{pub})$;
$\mathsf{k}_u \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pub})$;
$(c, \pi_a) \leftarrow \mathsf{EvalRequest}(st, \mathsf{k}_u, \mathsf{pub}, \mathsf{vk}^u_f)$
$(y, \pi) \leftarrow \mathsf{Eval}(c, f, \mathsf{sk}_f, \mathsf{pub})$
$(x_*, y_*, \pi_*) \leftarrow \mathcal{A}_2(\mathsf{sk}_f, f, c, y, \pi, \mathsf{pub})$;
If
$f(x_*) \neq y_*$ *and*
$\mathsf{VerifyResult}(y_*, \pi_*, \mathsf{vk}^u_f, \mathsf{k}_u, x_*, \mathsf{pub})$:
Then return 1*;*
Else return 0.

The advantage of the adversary $\mathcal{A}$ *against the UNF experiment is defined by:*

$$\mathsf{Adv}^{UNF}_{\Pi,\mathcal{A}}(\lambda) = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{UNF}_{\Pi,\mathcal{A}}(\lambda)\right].$$

A scheme Π *is UNF secure if the above advantage is negligible for any polynomial-time adversary* $\mathcal{A}$*.*

IV. PriVC: Privacy-Preserving Verifiable Computation

In the proposed system model, we assume that the cloud service provider is a semi-trusted party. We do not trust the server for computation on polynomial function. The system model consists of the following entities.

- Cloud Service Provider (CSP): CSP verifies authenticity of message and evaluates a function $f(x)$ over a user input m. The CSP also computes a proof of computation for $f(m)$.
- Users: Users are consumers of the company who gets services from CSP on their requests.
- Company: The company provides appropriate service in terms of evaluation of a function $f(x)$ with the help of CSP. The company also generates and securely distributes public and private parameters to all involved entities.

The PriVC scheme has following goals - (i) verifiable computation on encrypted data; (ii) secrecy of the prediction function $f(x)$; (iii) privacy-preserving user authentication; (iv) undeniability of user in a successful transaction. The privacy-preserving user authentication provides secure checking of user authenticity without revealing the actual identity of the user. Once a user sends a message to the CSP, the user cannot deny the transaction, and moreover, the CSP can prove that that particular user indeed did the transaction. The PriVC scheme works as follows. The company possesses a prediction function $f(x)$ and it hires CSP for computational power. The $f(x)$ is a confidential polynomial function and known to the company and CSP only. A user first gets registered with the company and receives a verification key and its public key. To avail the service, the user sends encryption of data m and the proof of authenticity to the CSP. To encrypt the data, we use a symmetric version of the extended DGHV scheme [16]. The CSP verifies the authenticity of the data and then generates an encrypted form of $f(m)$ along with the proof of computation. The user decrypts and verifies the computation of $f(m)$. Furthermore, a user cannot deny a valid transaction, and at the same time, the CSP cannot generate a valid transaction for any user. The system model of PriVC is depicted in Figure 2.

A. Construction of the PriVC

The Privacy-Preserving Verifiable Computation (PriVC) is a 8-tuple of algorithms (Setup, Init, KeyGen, EvalRequest, VerifyAuth, Eval, VerifyResult, VerifyTrans) as defined in Section 3 (Definition 6). The detailed working principle of each algorithm of the PriVC is as follows.

- $\mathsf{Setup}(\lambda)$: This algorithm generates a prime q and a multiplicative group G of order q. Let g be a generator of the group G. It randomly selects a secret key $sk_c \in \mathbb{Z}^\star_q$ for the company and sets $pk_c = g^{sk_c}$. It also selects a cryptographically secure hash function $H : \{0,1\}^\star \longmapsto \mathbb{Z}_q$. It sets $\mathsf{pub} = (q, pk_c, G, g, H)$.

- $\mathsf{Init}(f, \mathsf{pub})$: For the polynomial $f \in \mathbb{Z}_q[x]^n$, the company picks $(n+1)$ random numbers $\{\gamma_i \in \mathbb{Z}_q\}_{i=0}^n$ and sets $\Gamma(x) = \sum_{i=0}^n \gamma_i x^i$. For each user u in the system, it picks a secret sk_u randomly from $\mathbb{Z}^\star_q$ and sets $V_u(x) = sk_u^{-1}(\Gamma(x) - sk_c f(x))$. It sends the secret key $\mathsf{sk_f} = \Gamma(x)$ along with f to the CSP and the verification key $\mathsf{vk^u_f} = (V_u(x), pk_u = g^{sk_u})$ to user u.

- $\mathsf{KeyGen}(\lambda, \mathsf{pub})$: This algorithm is run by each user separately to generate extended DGHV private key. This algorithm picks a prime p in $[2^{\eta-1}, 2^\eta]$ where $\eta = \mathcal{O}(\lambda^2)$. It sets parameters $\rho = \mathcal{O}(\lambda)$ and $\gamma = \mathcal{O}(\lambda^5)$. The user's private key is $\mathsf{k}_u = p$.

- $\mathsf{EvalRequest}(m, \mathsf{k}_u, \mathsf{pub}, \mathsf{vk^u_f})$: This algorithm generates an authentic encrypted ciphertext query for the data m. We assume that the function f is a polynomial of degree n with coefficients $0 \leq a_i \leq q$ for $0 \leq i \leq n$. This algorithm first computes a tuple $c = (c_0, c_1, \ldots, c_n)$ as follows:

$$c_i = (m^i \bmod q) + s_i q + t_i p$$

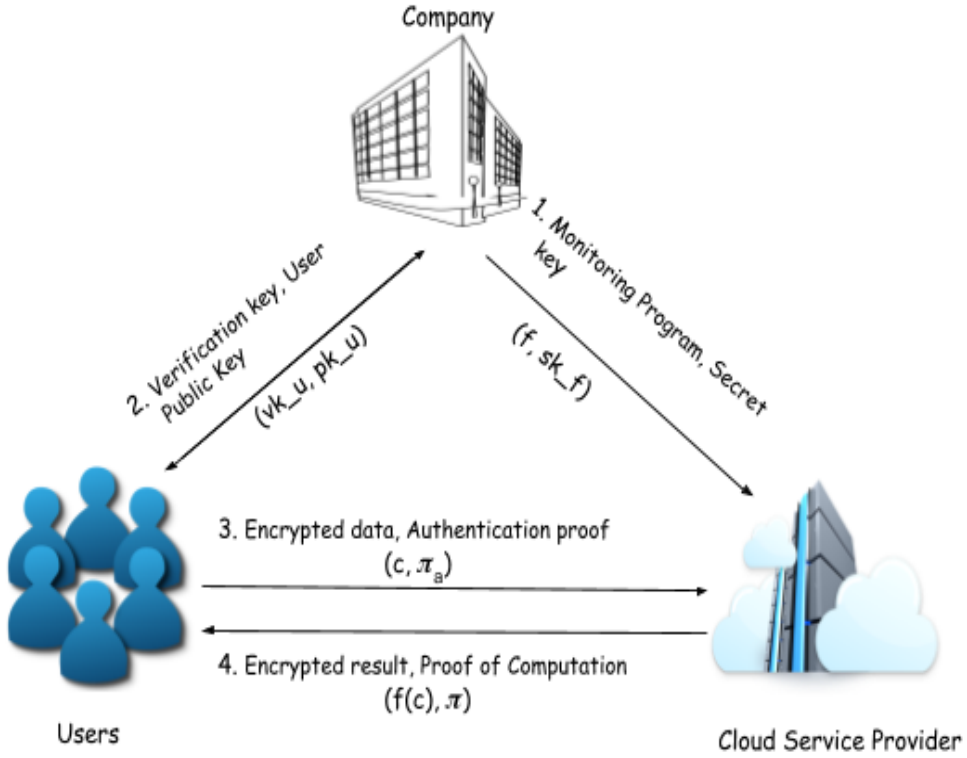

Fig. 2. Proposed scheme for monitoring system

where each $s_i \in (-2^\rho, 2^\rho)$ and $t_i \in (0, 2^\gamma/p)$ are randomly chosen integers. It then computes proof of authenticity $\pi_a = (R, S)$ as follows:

$$\theta = H(c)$$
$$R = V_u(\theta)$$
$$S = pk_u.$$

The user sends (c, π_a) to the CSP.

- VerifyAuth($c, \pi_a, \mathsf{sk_f}, f, \mathsf{pub}$): This algorithm verifies authenticity of c. The CSP computes $f(\theta)$, $\Gamma(\theta)$ where $\theta = H(c)$, and checks validity of the following equation:

$$S^R pk_c^{f(\theta)} \stackrel{?}{=} g^{\Gamma(\theta)}.$$

The algorithm outputs (c, π_a) if the above equation holds true; else, aborts.

- Eval($c, f, \mathsf{sk_f}, \mathsf{pub}$): This algorithm computes $f(c)$ and $\Gamma(c)$ as follows:

$$f(c) = \sum_{i=0}^{n} a_i c_i\,, \qquad \Gamma(c) = \sum_{i=0}^{n} \gamma_i c_i.$$

It sends $(y = f(c), \pi = \Gamma(c))$ to the user.

- VerifyResult($y, \pi, \mathsf{vk_f^u}, \mathsf{k_u}, m, \mathsf{pub}$): This algorithm computes $f(m)$ and $\Gamma(m)$ from (y, π) as follows:

$$f(m) = (y \bmod p) \bmod q$$
$$\Gamma(m) = (\pi \bmod p) \bmod q.$$

It then verifies following equation:

$$pk_u^{V_u(m)} pk_c^{f(m)} \stackrel{?}{=} g^{\Gamma(m)}.$$

- VerifyTrans($c, \pi_a, \mathsf{k_u}, \mathsf{vk_f^u}, \mathsf{pub}$): Using this algorithm, the user can verify whether a specific query (c, π_a) was sent by him/her. From c and $\pi_a = (R, S)$, if $S = pk_u$, then it computes $\theta = H(c)$. The user then verifies following equations:

$$R \stackrel{?}{=} V_u(\theta).$$

Correctness: We provide proof of correctness for equation in VerifyResult.

$$\begin{aligned} pk_u^{V_u(m)} pk_c^{f(m)} &= (g^{sk_u})^{sk_u^{-1}(\Gamma(m) - sk_c f(m))} (g^{sk_c})^{f(m)} \\ &= g^{\Gamma(m) - sk_c f(m)} (g^{sk_c})^{f(m)} \\ &= g^{\Gamma(m)}. \end{aligned}$$

B. Security Analysis

We first show that PriVC is IND-CFA secure. Note that for proof generation, we are using only one additional polynomial $\Gamma(x)$ and no other parameter. We show that this additional

polynomial $\Gamma(x)$ doesn't leak any additional information about the polynomial f.

Theorem 1. *$\forall n \in \mathbb{N}$, PriVC is unconditionally n-IND-CFA secure.*

Proof. Let $\mathcal{A}$ be an IND-CFA adversary for PriVC. We show that there exist a polynomial time algorithm $\mathcal{B}$ which can simulate the experiment $\mathsf{Exp}^{\text{n-IND-CFA}}_{\mathsf{PriVC},\mathcal{A}}(\lambda)$ to $\mathcal{A}$. The algorithm $\mathcal{B}$ works as follows:

- $\mathcal{B}$ picks $b \leftarrow \{0,1\}$.
- $\mathcal{B}$ generates $\mathsf{pub} = (q, pk_c, G, g, H) \leftarrow \mathsf{Setup}(\lambda)$ where $pk_c = g^{sk_c}$ and $sk_c \in \mathbb{Z}_q^\star$.
- $\mathcal{B}$ runs $(f_0, f_1, st) \leftarrow \mathcal{A}_1(\mathsf{pub}, n)$.
- $\mathcal{B}$ generates $(\mathsf{sk}_f, \mathsf{vk}_f^u) \leftarrow \mathsf{Init}(f_b, \mathsf{pub})$ where $mathsf sk_f = \Gamma(x) = \sum_{i=0}^{n} \gamma_i x^i$, $\mathsf{vk}_\mathsf{f}^\mathsf{u} = (V_u(x), pk_u)$, $pk_u = g^{sk_u}$ and $V_u(x) = sk_u^{-1}(\Gamma(x) - sk_c f_b(x))$.
- $\mathcal{B}$ generates $\mathsf{k_u} \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pub})$ where $\mathsf{k_u} = p$ is extended DGHV encryption key.
- $\mathcal{B}$ generates $(c, \pi_a) \leftarrow \mathsf{EvalRequest}(st, \mathsf{k_u}, \mathsf{pub}, \mathsf{vk}_f^u)$ where $c = (c_0, c_1, ..., c_n)$, c_i is encryption of x^i and $\pi_a = (V_u(H(c)), pk_u)$.
- $\mathcal{B}$ runs $b_\star \leftarrow \mathcal{A}_2^{\mathcal{O}_{CFA}(\cdot)}(\mathsf{k_u}, \mathsf{vk}_\mathsf{f}^\mathsf{u}, \mathsf{pub}, f_0, f_1, c)$. To simulate the oracle $\mathcal{O}_{CFA}(\cdot)$ on c to $\mathcal{A}$ on c, $\mathcal{B}$ picks $r_1, r_2 \leftarrow \{0,1\}^\star$ and set $f' = f_b + r_1(f_0 - f_1)$ and $sk'_f = \Gamma'(x) = sk_f + r_2(f_0 - f_1)$ and computes $y = f'(c)$ and $\pi = \Gamma'(c)$. It returns (y, π).
- Finally, $\mathcal{B}$ outputs $b_\star$.

We note that if $f_0(st) = f_1(st)$ then we have $f_b(st) = (f'(st) \mod p) \mod q$ and $\Gamma(st) = (\Gamma'(st) \mod p) \mod q$. Finally, $pk_u^{V_u(st)} pk_c^{f_b(st)} = (g^{sk_u})^{sk_u^{-1}(\Gamma(st) - sk_c f_b(st))} (g^{sk_c})^{f_b(st)} = g^{\Gamma(st)}$.

We conclude that the n-IND-CFA experiment is successfully simulated for $\mathcal{A}$. Hence, the success of the adversary $\mathcal{A}$ is equal to the randomly guessing the value b. This gives

$$\Pr\left[1 \leftarrow \mathsf{Exp}^{\text{n-IND-CFA}}_{\mathsf{PriVC},\mathcal{A}}(\lambda)\right] = \frac{1}{2}$$

and the $\mathsf{Adv}^{\text{n-IND-CFA}}_{\mathsf{PriVC},\mathcal{A}}(\lambda)$ is negligible. Hence, the PriVC scheme is n-IND-CFA secure. □

We show that our scheme is unforgeable under discrete logarithm assumption.

Theorem 2. *The proof of computation in PriVC is UNF-secure under the discrete logarithm assumption.*

Proof. We show that if an adversary $\mathcal{A}$ can successfully forge the proof of computation then it can break the discrete logarithm assumption by computing $\log_g pk_c$. Let $(x_\star, y_\star, \pi_\star)$ be a forged result with $\mathsf{VerifyResult}(y_\star, \pi_\star, \mathsf{vk}_f^u, \mathsf{k_u}, x_\star, \mathsf{pub}) = 1$ and $f(x_\star) \neq y_\star$.

$\mathcal{A}$ can compute $\Gamma' = (\pi_\star \mod p) \mod q$. We have

$$pk_u^{V_u(x_\star)} pk_c^{y_\star} = g^{\Gamma'} \quad \text{and} \quad pk_u^{V_u(x_\star)} pk_c^{f(x_\star)} = g^{\Gamma(x_\star)}.$$

From this, the adversary $\mathcal{A}$ can deduce that

$$pk_c = g^{\frac{\Gamma' - \Gamma(x_\star)}{y_\star - f(x_\star)}}.$$

Finally, $\mathcal{A}$ computes $\log_g pk_c = \dfrac{\Gamma' - \Gamma(x_\star)}{y_\star - f(x_\star)}$. Based on discrete logarithm assumption, there cannot exist an adversary $\mathcal{A}$ such that the advantage $\mathsf{Adv}^{UNF}_{\mathsf{PriVC},\mathcal{A}}(\lambda)$ is non-negligible. □

User non-repudiation: Assume that $(c, \pi_a = (R, S))$ be a valid query for a user u and the server creates a fake query $(c', \pi'_a = (R', S))$ for the user u. If $H(c) = H(c')$, then the server can set $\pi'_a = \pi_a$ and (c', π'_a) becomes a valid query for the user u. Since the hash function H is assumed to be a collision resistant, it is computationally difficult to find c' such that $H(c) = H(c')$. For any (c', R'), the query (c', π'_a) is valid for the user u if only if $V_u(H(c')) = R'$. Since the verification function $V_u(x)$ is not public and $V_u(x) \in \mathbb{Z}_q[x]$, the probability of $V_u(H(c')) = R'$ is $1/q$. For sufficient value of q, this probability is negligible.

The algorithm VerifyAuth ensures that an unauthorized user cannot access the service. Since verification key's of each user is kept secret, the probability that an unauthorized user generates a valid query (c, π_a) where $\pi_a = (R, S)$ is equal to the probability of $R = V_u(H(c))$ for any user u. Since $V_u(x)$ is in $\mathbb{Z}_q[x]$, the probability of $R = V_u(H(c))$ for any random R is $1/q$.

Security of Encryption: For encryption of user data, we are using a symmetric version of the extended DGHV scheme proposed by Pedro *et al.* [16]. They have proved that their scheme is semantically secure under the assumption of approximate GCD problem. However, we note that there are two attacks on DGHV scheme, and we show that the extended DGHV scheme is secure under both the attacks. First is key recovery attack [18] and another is lattice attack [19]. In the key recovery attack, the adversary has access to the decryption oracle, and it can successfully find the key p after a few communications with the decryption oracle. The attack was successful because B is equal to 2 in the DGHV scheme, and therefore, the adversary was able to reduce the search space in half after every communication with the decryption oracle. In our scheme, B is a prime q of $\mathcal{O}(\lambda)$ bits. Therefore, the adversary can reduce the search space down to the size of an order of 2^λ, and it still has to do brute force for 2^λ many numbers. Therefore, the extended DGHV scheme is secure under a key recovery attack. In the lattice attack, the adversary considers a vector of several ciphertexts together and reduces the search space of vectors of plaintexts to the size of B^2. For $B = 2$, this is just 4, and if the plaintexts are binary, then it is easy to find the correct message out of 4. We have used $B = q$ where q is $\mathcal{O}(\lambda)$ bit prime. Therefore, the lattice attack can reduce the search space of vectors of plaintexts to the size of $B^2 \approx 2^{2\lambda}$. Hence, the extended DGHV scheme is secure under lattice attack as well.

We note that if a user collude with the CSP, the CSP can provide the polynomial to the user. This problem is inherent and cannot be prevented. Client can share verification key with the server but it does not help the server to forge the proof of computation for other users as each user has different verification key.

TABLE I
PERFORMANCE LEVEL FOR VARIOUS TEST INSTANCES

Instance	EvalRequest	VerifyAuth	Eval	VerifyResult	VerifyTrans
Tiny	3.25	0.176	0.169	0.154	0.122
Small	20.8	0.645	0.884	0.739	0.587
Medium	125	3.13	4.92	4.48	3.08
Large	651	14.2	32.3	25.3	14.1

All times in the Table I are in milliseconds.

TABLE II
PARAMETERS USED FOR VARIOUS TEST INSTANCES

Instance	λ^{a}	ρ^{b}	η^{b}	γ^{b}
Tiny	42	27	1026	150000
Small	52	41	1558	830000
Medium	62	56	2128	4200000
Large	72	71	2698	19350000

[a] λ denotes security level.
[b] ρ, η and γ denotes bit-size of parameters in extended DGHV.

V. Experimental Results

To check the practicality of the scheme, we have implemented the PriVC scheme with realistic parameters.

In this section, we describe the implementation of PriVC and parameters selection. To prevent key recovery attack and lattice attacks on extended DGHV scheme, we must have $\rho = \lambda$, $\eta = \mathcal{O}(\lambda^2)$, $\gamma = \mathcal{O}(\lambda^5)$. Throughout our experiment, the degree of the polynomial $f(x)$ and $\Gamma(x)$ is 10. The coefficients of all three polynomials are randomly selected from $\mathbb{Z}_q$ where q is a 256 bit integer.

We considered four test instances with different security level as described in the Table II. We have implemented PriVC scheme using SageMath version 8.1 on a quad-core desktop computer with Intel Core i5 − 6500 at 3.2 GHz and 4 GB RAM.

Table I summarizes the performance of our implementation of PriVC. We considered four instances for our experiment, namely Tiny, Small, Medium, and Large. For each instance, we use parameters as described in the Table II. The time in the Table I represents the running time (in milliseconds) of a single query averaged over 10000 iterations. When we increase the security level, the size of the encrypted query increases, this leads to more computational cost during query generation, and function evaluation. The encrypted query generation depends on the extended DGHV parameters. The size of the parameters for the extended DGHV scheme increases as we increase the security level. As expected, the query generation time increases as we increase the security of the underlying encryption scheme. Even for high-security level, Large instance, the computational cost for each process is a fraction of a second.

The result verification involves mainly 3 exponentiation and polynomial evaluation. The algorithm VerifyResult first decrypts the result and then verify it using the verification equation. The time for VerifyResult in Table I is the total time required for decrypting the result and then verifying it. The time required for result verification is reasonably small compared to actual polynomial evaluation time. A quick look at the execution times in Table I shows that our scheme is efficient and suitable for the practical purpose.

VI. Conclusion

In this paper, we provide a construction of provable private polynomial evaluation scheme. This scheme allows a company to delegate computation of a secret polynomial to the CSP in such a way that a user can verify the computation. The CSP can verify user authenticity in a privacy-preserving manner. Moreover, once the transaction is done, the user cannot deny it later. Using formal security models for IND-CFA and UNF, we prove that our scheme PriVC is IND-CFA and UNF secure. We implemented the proposed scheme using SageMath version 8.1 and observe that the result verification step takes less time compared to an actual polynomial evaluation done by the CSP.

Acknowledgment

This research was conducted with the support of the Indo-French Centre for the Promotion of Advanced Research (IFCPAR) and the Center Franco-Indien Pour La Promotion De La Recherche Avancée (CEFIPRA) through the project DST/CNRS 2015-03 under DST-INRIA-CNRS Targeted Programme.

References

[1] J. K. O'herrin, N. Fost, K. A. Kudsk, "Health insurance portability accountability act (HIPPA) regulations: effect on medical record research", Annals of Surgery, vol. 239(6), pp. 772–778, Jun 2004.

[2] R. Gennaro, C. Gentry, B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers", In proceedings of the 30th annual conference on Advances in Cryptology (CRYPTO), Santa Barbara, CA, USA, pp. 465–482, 2010.
[3] B. Parno, M. Raykova, V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption", In proceedings of the 9th International Conference on Theory of Cryptography (TCC), Taormina, Sicily, Italy, pp. 422–439, 2012.
[4] B. Parno, J. Howell, C. Gentry, M. Raykova, "Pinocchio: Nearly practical verifiable computation", In Proceedings of IEEE Symposium on Security and Privacy, Berkeley, CA, pp. 238–252, 2013.
[5] C. Costello, "Geppetto: Versatile verifiable computation", In proceedings of IEEE Symposium on Security and Privacy, San Jose, CA, pp. 253–270, 2015.
[6] J. Ye, H. Zhang, C. Fu, "Verifiable delegation of polynomials", International Journal of Network Security, vol. 18(2), pp. 283–290, 2016.
[7] M. Backes, D. Fiore, R. M. Reischuk, "Verifiable delegation of computation on outsourced data", In proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (CCS), Berlin, Germany, pp. 863–874, 2013.
[8] D. Fiore, R. Gennaro, "Publicly verifiable delegation of large polynomials and matrix computations, with applications", In proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS), Raleigh, NC, USA, pp. 501–512, 2012.
[9] M. Naor, B. Pinkas, "Oblivious transfer and polynomial evaluation", In proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing (SoTC), Atlanta, Georgia, USA, pp. 245–254, 1999.
[10] A. Kate, G. M. Zaverucha, I. Goldberg, "Constant-size commitments to polynomials and their applications", In proceedings of 16th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), Singapore, pp. 177–194, 2010.
[11] L. Guo, Y. Fang, M. Li, P. Li, "Verifiable privacy-preserving monitoring for cloud-assisted mHealth systems", In proceedings of the IEEE Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, pp. 1026–1034, 2015.
[12] H. Gajera, S. Naik, M. L. Das, "On the security of "Verifiable Privacy-Preserving Monitoring for Cloud-Assisted mHealth Systems"", In proceedings of the 12th International Conference on Information System Security (ICISS), Jaipur, India, pp. 324–335, 2016.
[13] N. Howgrave-Graham, "Approximate integer common divisors", In: J. H. Silverman (ed.) Cryptography and Lattices, LNCS, vol. 2146, pp. 51–66, 2001.
[14] W. Diffie, M. E. Hellman, "New directions in cryptography", In IEEE Transactions on Information Theory, vol. 22(6), pp. 64–654, 1976.
[15] D. Micciancio, "A first glimpse of cryptographys holy grail", In Communications of the ACM, vol. 53(3), pp. 96–96, March 2010.
[16] P. S. Pisa, M. Abdalla, O. C. M. B. Duarte, "Somewhat homomorphic encryption scheme for arithmetic operations on large integers", In Global Information Infrastructure and Networking Symposium (GIIS), Choroni, Venezuela, pp. 1–8, 2012.
[17] M. Van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, "Fully homomorphic encryption over the integers", In proceedings of 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Monaco and Nice, French Riviera, pp. 24–43, 2010.
[18] M. Chenal, Q. Tang, "On key recovery attacks against existing somewhat homomorphic encryption schemes", In proceedings of 3rd International Conference on Cryptology and Information Security in Latin America (LATINCRYPT), Florianpolis, Brazil, pp. 239–258, 2014.
[19] T. Lepoint, M. Tibouchi, "Cryptanalysis of a (somewhat) additively homomorphic encryption scheme used in PIR", In proceedings of International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico, pp. 184–193, 2015.
[20] B. Xavier, M. L. Das, H. Gajera, D. Gerault, M. Giraud, P. Lafourcade, "Verifiable private polynomial evaluation", In proceedings of 11th International Conference on Provable Security (ProvSec), Xi'an, China, pp. 487–506, 2017.