Verifiable Private Polynomial Evaluation

Xavier Bultel[1], Manik Lal Das[2], Hardik Gajera[2], David Gérault[1], Matthieu Giraud[1], and Pascal Lafourcade[1]

[1] Université Clermont Auvergne, CNRS, LIMOS, Clermont-Ferrand, France
firstname.lastname@uca.fr
[2] DA-IICT, Gandhinagar, India
{maniklal,kidrah123}@gmail.com

Abstract. Delegating the computation of a polynomial to a server in a verifiable way is challenging. An even more challenging problem is ensuring that this polynomial remains hidden to clients who are able to query such a server. In this paper, we formally define the notion of *Private Polynomial Evaluation* (PPE). Our main contribution is to design a rigorous security model along with relations between the different security properties. We define *polynomial protection* (PP), *proof unforgeability* (UNF), and *indistinguishability against chosen function attack* (IND-CFA), which formalizes the resistance of a PPE against attackers trying to guess which polynomial is used among two polynomials of their choice. As a second contribution, we give a cryptanalysis of two PPE schemes of the literature. Finally, we design a PPE scheme called PIPE and we prove that it is PP-, UNF- and IND-CFA-secure under the decisional Diffie-Hellman assumption in the random oracle model.

1 Introduction

Mathematical models are powerful tools that are used to make predictions about a system's behaviour. The idea is to collect a large set of data for a period of time and use it to build a function predicting the evolution of the system in the future. This topic has many applications, for instance, meteorology or economics. It can be used to predict the weather or the behaviour of stock exchange.

Consider a company that collects and stores a very large set of data, for example about the state of the soil, such as humidity, acidity, temperature and mineral content. Using it, it computes some function that predicts the state of the soil for next years. The clients are farmers who want to anticipate the state of the soil during the sowing periods to determine how much seeds to buy and when to plant them. The company gives its client access to the prediction function through a cloud server. A paying client can then interact with the server to evaluate the function on his own data. For economic reasons, the company does not want the clients to be able to recover the prediction function. Moreover, the clients do not trust the server: it might be corrupted to produce incorrect results. Hence, the server should provide a proof that its output is correct with regards to the secret prediction function. A similar scenario was studied in [GFLL15],

where a server receives medical data collected by sensors worn by the users, and provides the users with an evaluation of their health status. More precisely, the company defines a polynomial f which returns meaningful information, such as potential diseases. Then, it uploads this polynomial to the server, and sells to the end users the ability to query that function with their own medical data.

The underlying problem is how to delegate computations on a secret polynomial function to a server in a verifiable way. By *secret* we mean that no user should be able to retrieve the polynomial used by the server. By *verifiable* we mean that the server must be able to prove the correctness of its computation. To solve this problem, we propose the *Private Polynomial Evaluation* (PPE) primitive, which ensures that: (i) the polynomial f is protected as much as possible, and (ii) the user is able to verify the result given by the server.

Figure 1 illustrates a PPE scheme where x is the user data and $f(x)$ is the evaluation of the data by the function f of the company. Moreover, the proof π sent by the server and the verification key vk sent by the company allow the user to verify the correctness of the delegated computation.

Consider a company using a PPE scheme for prediction functions. An attacker wants to guess which prediction function is used by the company. Assume this attacker gains access to some of the data used to build the prediction function, for instance by corrupting a technician. Thus, the attacker can build several prediction functions by using different mathematical models and the collected data, and try to distinguish which of these functions is used by the company. Intuitively, in a secure PPE scheme, this task should be as hard as if the server only returned $f(x)$, and no additionnal information for verification. We formalise this notion and design a PPE scheme having this security property.

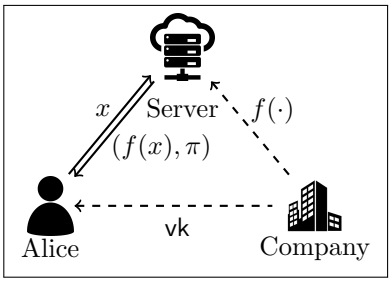

Fig. 1: Illustration of a PPE scheme.

Contributions:

- We give a cryptanalysis of two PPE schemes, the first one presented by Guo *et al.* [GFLL15] and the second one presented by Gajera *et al.* [GND16]. Our attack allows an adversary to recover the secret polynomial in a single query.
- Our main contribution is to provide a formal definition and security framework for PPE schemes. We define two one-way notions, *Weak Polynomial Protection* (WPP) and *Polynomial Protection* (PP), stating that a user limited to k queries cannot recover the polynomial, where k is the degree of the polynomial. Additionally, we define IND-CFA which formalises the idea that no adversary can guess which of two polynomials of his choice is used. In essence, the proof of a correct computation should not reveal any infor-

mation about the polynomial. We finally study the relations between these notions.
– We design PIPE (for Private IND-CFA Polynomial Evaluation), an efficient IND-CFA-secure PPE scheme. This scheme combines the Verifiable Secret Sharing introduced by Feldman [Fel87] and the ElGamal encryption scheme in order to achieve verifiability and IND-CFA security. We also formally prove its security under the DDH assumption in the random oracle model.

Related works: Verifiable Computation (VC) refers to the cryptographic primitives where an untrusted server can prove the correctness of its output. It was introduced in [GGP10]. The aim of a such primitive is to allow a client with limited computational power to delegate difficult computations. Primitives where everyone can check the correctness of the computation are said to be *publicly verifiable* [PRV12]. This subject has led to a dense literature [PST13,CRR12,FG12] [CKKC13,PHGR13]. In 2012, Canetti *et al.* [CRR12] proposed formal security models for VC. Fiore and Gennaro [FG12] propose a scheme for polynomial evaluations and matrix computations. Unlike our paper, these works consider that the polynomial used by the server is public.

To the best of our knowledge, four papers study how to hide the function used by the server [GFLL15,GND16,KZG10,NP99]. Kate *et al.* define a primitive called *commitment to polynomials* (CTP) [KZG10]. In this primitive, a user commits to a hidden polynomial f and reveals some points (x, y) together with a proof that $f(x) = y$. The user can open the commitment *a posteriori* to reveal the polynomial. CTP is close to PPE: the verification key in a PPE scheme can be viewed as a commitment in a CTP scheme, the main difference is that this verification key is computed by a trusted party (the company) and the points are evaluated by an untrusted party (the server). The authors formalise the hardness of guessing the polynomial knowing less than k points. In this model, the polynomial is randomly chosen, then they does not consider the case where the adversary tries to distinguish the committed polynomial between two chosen polynomials as in our IND-CFA model. Moreover, Kate *et al.* design two CTP schemes in [KZG10]. The first one is not IND-CFA since the commitment algorithm is deterministic. We prove that the second scheme is IND-CFA-secure in the extended version [BDG+17]. Moreover, we show that our scheme PIPE can be used as a CTP scheme, and we compare it to the scheme of Kate *et al.*. We show that our scheme solves an open problem described by Kate *et al.*: designing a scheme that is secure under a weaker assumption than t-SDDH.

Independently of Kate *et al.* [KZG10], Guo *et al.* [GFLL15] propose a scheme with similar security properties to delegate the computation of a secret health related function on the users' health record. The polynomials are explicitly assumed to have low coefficients and degree, which greatly reduces their randomness. However, the authors give neither security models nor proof. Later, Gajera *et al.* [GND16] show that any user can guess the polynomial using the Lagrange's interpolation on several points. They propose a scheme where the degree k is hidden and claim that it does not suffer from this kind of attack. We show that hiding the degree k is useless and that no scheme can be secure when user query

more than k points to the server. Moreover, we give a cryptanalysis on these both schemes which requires only one query to the server. To the best of our knowledge, we present the first security model for *Indistinguishability Against Chosen Function Attack* (IND-CFA).

Finally, there has been lots of work done on a similar but slightly different topic, Oblivious Polynomial Evaluation (OPE), introduced by Naor and Pinkas [NP99]. In OPE, there are two parties. One party A holds a polynomial f and another party B holds an element x. The aim of OPE is that the party B receives $f(x)$ in such a way that A learns nothing about x and B learns nothing about f, except $f(x)$. Researchers have studied OPE extensively and shown that it can be used to solve various cryptographic problems, such as set membership, oblivious keyword search, data entanglement, set-intersection and more [FIPR05,FNP04,LP02]. Despite the similarities between OPE and PPE, they are different in nature. In particular, OPE does not consider the verifiability of $f(x)$, whereas it is a crucial point in PPE. Additionally, in a PPE, the requirement that the server does not learn anything about x is relaxed. In our scheme, the major contribution to computational cost is due to computation of the proof on server side and verification of computation on user side. Since OPE doesn't consider verifying computation, we feel that it would not be fair to compare the performances.

Outline: In the next section we recall the cryptographic notions used in this paper. In Section 3, we show how to break schemes proposed by Guo *et al.* [GFLL15] and by Gajera *et al.* [GND16]. In Section 4, we propose security models for PPE schemes. Finally, in Section 5, we present our PPE scheme PIPE and we prove that it is IND-CFA-secure before concluding.

2 Cryptographic Tools

We start by recalling the basic cryptographic assumptions used in this paper. In the following, we denote by $\text{POLY}(\lambda)$ the set of probabilistic polynomial time algorithms with respect to the security parameter λ.

Definition 1 (Discrete Logarithm assumption [DH76]). *Let p be a prime number generated according to a security parameter $\lambda \in \mathbb{N}$. Let G be a multiplicative group of order p, and $g \in G$ be a generator. The* discrete logarithm assumption *(DL) in (G, p, g) states that there exists a negligible function ϵ such that for all $x \xleftarrow{\$} \mathbb{Z}_p^*$ and $\mathcal{A} \in \text{POLY}(\lambda)$: $\mathsf{Pr}[x' \leftarrow \mathcal{A}(g^x) : x = x'] \leq \epsilon(\lambda)$*

Definition 2 (Decisional Diffie-Hellman assumption [Bon98]). *Let p be a prime number generated according to a security parameter $\lambda \in \mathbb{N}$. Let G be a multiplicative group of order p, and $g \in G$ be a generator. The* Decisional Diffie-Hellman assumption *(DDH) in (G, p, g) states that there exists a negligible function ϵ such that for all $(x, y, z) \leftarrow (\mathbb{Z}_p^*)^3$ and $\mathcal{A} \in \text{POLY}(\lambda)$:*

$$|\mathsf{Pr}[b \leftarrow \mathcal{A}(g^x, g^y, g^z) : b = 1] - \mathsf{Pr}[b \leftarrow \mathcal{A}(g^x, g^y, g^{x \cdot y}) : b = 1]| \leq \epsilon(\lambda)$$

In the following, we recall definition and security requirements of public key cryptosystems.

Definition 3 (Public Key Encryption). *A Public Key Encryption (PKE) scheme is defined by three algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *as follows:*

$\mathsf{Gen}(\lambda)$: *It returns a public/private key pair* $(\mathsf{pk}, \mathsf{sk})$.
$\mathsf{Enc}_{\mathsf{pk}}(m)$: *It returns the ciphertext* c *of the message* m.
$\mathsf{Dec}_{\mathsf{sk}}(c)$: *It returns the plaintext* m *from the ciphertext* c.

A PKE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *indistinguishable under chosen-plaintext attack* (IND-CPA) if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the difference between $\frac{1}{2}$ and the probability that $\mathcal{A}$ wins the IND-CPA experiment presented in Figure 2 is negligible in λ. The oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{LR}_b(\cdot,\cdot))$ takes (m_0, m_1) as input and returns $\mathsf{Enc}_{\mathsf{pk}}(m_b)$. The standard definition of CPA experiment allows the adversary to call this oracle only one time. However, Bellare *et al.* [BBM00] prove that the two definitions of CPA security are equivalent using a hybrid argument. For instance, the ElGamal encryption is IND-CPA.

$\mathbf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\lambda)$:
$b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\lambda)$;
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}_{\mathsf{pk}}(\mathsf{LR}_b(\cdot,\cdot))}(\lambda, \mathsf{pk})$
return $(b = b')$

Fig. 2: IND-CPA experiment [BBM00].

Definition 4 (ElGamal Encryption [ElG85]). *The* ElGamal *PKE scheme is defined as follows:*

$\mathsf{Gen}(\lambda)$: *It returns* $\mathsf{pk} = (G, p, g, h)$ *and* $\mathsf{sk} = x$ *where* G *is a multiplicative group of prime order* p, g *is a generator of* G, $h = g^x$ *and* x *is uniform in* $\mathbb{Z}_p^*$.
$\mathsf{Enc}_{\mathsf{pk}}(m)$: *It returns* $(c, d) = (g^r, h^r \cdot m)$ *where* r *is randomly chosen in* $\mathbb{Z}_p^*$.
$\mathsf{Dec}_{\mathsf{sk}}((c, d))$: *It returns* $m = d \cdot c^{-x}$.

A zero-knowledge proof (ZKP) allows a prover knowing a witness to convince a verifier that a statement s is in a given language without leaking any information except s. We recall the definition of a non-interactive ZKP.

Definition 5 (NIZKP [FS87]). *A* non-interactive ZKP *(NIZKP) for a language* $\mathcal{L}$ *is a couple of algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *such that:*

$\mathsf{Prove}(s, w)$: *It outputs a proof* π *that* $s \in \mathcal{L}$ *using the witness* w.
$\mathsf{Verify}(s, \pi)$: *It checks whether* π *is a valid proof that* $s \in \mathcal{L}$ *and outputs a bit.*

A NIZKP proof verifies the following properties:

Completeness *For any statement* $s \in \mathcal{L}$ *and the corresponding witness* w, *we have that* $\mathsf{Verify}(s, \mathsf{Prove}(s, w)) = 1$.
Soundness *There is no polynomial time adversary* $\mathcal{A}$ *such that* $\mathcal{A}(\mathcal{L})$ *outputs* (s, π) *such that* $\mathsf{Verify}(s, \pi) = 1$ *and* $s \notin \mathcal{L}$ *with non-negligible probability.*

Zero-knowledge. *A proof* π leaks no information, i.e. *there exists a PPT algorithm* Sim *(called the* simulator*) such that outputs of* $\mathsf{Prove}(s, w)$ *and the outputs of* $\mathsf{Sim}(s)$ *follow the same probability distribution.*

We use the NIZKP given by Chaum and Pedersen [CP93] to prove the equality of two discrete logarithms. Let G be a multiplicative group, the language is the set of all statements $(g_1, h_1, g_2, h_2) \in G^4$ such that $\log_{g_1}(h_1) = \log_{g_2}(h_2) = x$.

Definition 6 (LogEq [CP93]). *Let* G *be a multiplicative group of prime order* p *and* H *be a hash function,* $\mathcal{L}$ *be the set of all* $(g_1, h_1, g_2, h_2) \in G^4$ *where* $\log_{g_1}(h_1) = \log_{g_2}(h_2)$*. We define the NIZKP* $\mathsf{LogEq} = (\mathsf{Prove}, \mathsf{Verify})$ *for* $\mathcal{L}$ *as follow:*

$\mathsf{Prove}((g_1, h_1, g_2, h_2), w)$**:** *Using the witness* $w = \log_{g_1}(h_1)$*, it picks* $r \xleftarrow{\$} \mathbb{Z}_p^*$*, computes* $A = g_1^r$*,* $B = g_2^r$*,* $z = H(A, B)$ *and* $\omega = r + w \cdot z$*. It outputs* $\pi = (A, B, \omega)$*.*

$\mathsf{Verify}((g_1, h_1, g_2, h_2), \pi)$**:** *Using* $\pi = (A, B, \omega)$*, it computes* $z = H(A, B)$*. If* $g_1^\omega = A \cdot h_1^z$ *and* $g_2^\omega = B \cdot h_2^z$ *then it outputs* 1*, else it outputs* 0*.*

LogEq *is unconditionally complete, sound and zero-knowledge in the ROM.*

We recall Lagrange's interpolation formula to find the single polynomial f of degree at most k from $k+1$ points (x_i, y_i) such that $f(x_i) = y_i$.

Definition 7 (Lagrange's interpolation). *Let* k *be an integer and* F *be a field. For all* $i \in \{0, \ldots, k\}$*, let* $(x_i, y_i) \in F^2$ *such that for all* $i_1, i_2 \in \{0, \ldots, k\}$*,* $x_{i_1} \neq x_{i_2}$*. There exists one and only one polynomial* f *of degree at most* k *such that for all* $i \in \{0, \ldots, n\}$*,* $f(x_i) = y_i$*. This polynomial is given by Lagrange's interpolation formula:*

$$f(x) = \sum_{i=0}^{k} \left(y_i \cdot \prod_{j=0, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \right) .$$

In the following, we denote the set of polynomials with coefficients in the field F by $F[X]$ and we denote the set of all $f \in F[X]$ of degree k by $F[X]_k$.

3 Cryptanalysis of [GFLL15] and [GND16]

We start by presenting the inherent limitation of PPE schemes, then we explain how to break those presented by Guo *et al.* [GFLL15] and by Gajera *et al.* [GND16].

3.1 Inherent Limitation

In the scheme [GFLL15], the degree k of the polynomial f is public. Gajera *et al.* [GND16] use it to mount an attack: a user queries $k+1$ points to guess the

polynomial using Lagrange's interpolation. To fix this weakness, they propose a scheme where k is secret. However, any user can guess k and f after $k+1$ interactions with the server. To do so, the attacker chooses an input x_0 and sends it to the server. He receives y_0 and computes the polynomial f_0 of degree 0 using Lagrange's interpolation on (x_0, y_0). Next, the attacker chooses a second and a different input x_1 and asks $y_1 = f(x_1)$ to the server. He computes the polynomial f_1 of degree 1 using Lagrange's interpolation on $\{(x_0, y_0), (x_1, y_1)\}$. By repeating this process until the interpolation gives the same polynomial $f_i = f_{i+1}$ for two consecutive iterations, he recovers the degree and the polynomial. This problem is an inherent limitation of PPE schemes and was already considered in the security model of Kate *et al.* [KZG10]. Thus, to preserve the protection of the polynomial, the server must refuse to evaluate more than k points for each client and we must assume that clients do not collude to collect more than k points.

3.2 Cryptanalysis of [GFLL15] and [GND16]

In addition to the protection of f, the scheme [GFLL15] requires that the user's data is encrypted for the server. More formally, the user uses an encryption algorithm to compute $x' = \mathsf{Enc}_k(x)$ and sends this cipher to the server which returns y'. Then, the user computes $y = \mathsf{Dec}_k(y')$ such that $y = f(x)$ where f is the secret polynomial. The encryption scheme is based on the discrete logarithm assumption. The decryption algorithm works in two steps: first the user computes a value h such that $h = g^{f(x)}$ where g is a generator of a multiplicative group of large prime order n, next he computes the discrete logarithm of h in base g using *Pollard's lambda method* [Pol78]. The authors assume that the size of $f(x)$ is reasonable: more formally, they define a set of possible inputs $\mathcal{X}$ and $M \in \mathbb{N}$ such that $\forall x \in \mathcal{X}, 0 \leq f(x) < M$. The authors assume that the users can efficiently perform Pollard's lambda algorithm on any $h = g^y$ where $y < M$. Actually, for practical reasons, since $h = g^{f(x) \mod n}$ and $\log_g(h) = f(x)$, we assume that $0 \leq f(x) < n$ for any input x of reasonable size, *i.e.* $x \ll n$. Hence, the authors of [GFLL15] consider f as a positive polynomial in $\mathbb{Z}$ with sufficiently small coefficients.

It is easy to evaluate a small M' such that $M' > M$ by choosing M' such that Pollard's lambda algorithm on $g^{M'}$ is computable by a powerful server but is too slow for a practical application. For example, if Pollard's lambda algorithm takes less than one minute for the server but more than one hour for the user's computer, we can assume that $M' > M$ and attacks that are polynomial in M' are practical. To sum up, the user has the following tools:

- $M' \in \mathbb{N}$ such that $\forall x \in \mathcal{X}$, $0 \leq f(x) < M'$ and such that algorithms that require $p(M')$ operations (where p is a polynomial) are *easily computable*.
- A server which returns $y = f(x)$ for any input x. This server can be used at most k times where k is the degree of the polynomial.

Finally, note that the authors assume that $0 \leq f(x)$ for any x and that $\mathcal{X} \subset \mathbb{N}$. We show that any user can guess the secret polynomial during his first interaction with the server. We first prove the following two properties.

Property 1. For any polynomial $f \in \mathbb{Z}[X]$ and any integers x and y, there exists $P \in \mathbb{Z}$ such that

$$f(x+y) = f(x) + y \cdot P \ .$$

Proof. Seen as a polynomial in y, $f(x+y) - f(x)$ has a root at $y = 0$. By the Factor Theorem y divide $f(x+y) - f(x)$. Hence, there exists $P \in \mathbb{Z}$ such that $f(x+y) - f(x) = y \cdot P$, *i.e.* $f(x+y) = f(x) + y \cdot P$.

Note that for any positive integers a and b such that $a < b$, we have $a \mod b = a$. Then, we can deduce the following property from Property 1.

Property 2. For any polynomial $f \in \mathbb{Z}[X]$ and any integers x and y such that $0 \leq f(x) < y$ and $0 \leq f(x+y)$, it holds that

$$f(x+y) \mod y = f(x) \ .$$

Proof. From the previous property, we have $f(x+y) = f(x)+y \cdot P$, where P is an integer. Assume $P < 0$, we define $P' = -P > 0$, then $f(x+y) = f(x)-y \cdot P' \geq 0$. Hence we have $f(x) \geq y \cdot P' > f(x) \cdot P'$.

- If $0 < f(x)$ then we deduce $1 = f(x)/f(x) > P'$ and $1 > P'$.
- If $f(x) = 0$ then $0 \geq y \cdot P' > 0$.

In both cases, we obtain a contradiction. We conclude that $0 \leq P$. Finally, we deduce $f(x+y) \mod y = f(x) + y \cdot P \mod y = f(x)$.

Our attack on [GFLL15] works as follows. The attacker chooses a vector of k integers $(x_1, x_2, \ldots, x_k) \in \mathbb{N}^k$ such that, for all $0 < i \leq k, x'_i = \sum_{j=1}^{i} x_j$ where $x'_i \in \mathcal{X}$.

For the sake of clarity, we show to begin with the attack in the case where $\{1, \ldots, k\} \subset \mathcal{X}$. Thus the attacker chooses the vector $(x_1, x_2, \ldots, x_k) = (1, 1, \ldots, 1)$ and sends $x = k + M'$ to the server that returns the encryption of $y = f(x)$. Pollard's lambda algorithm complexity [Pol78] on M' is $O(M'^{1/2})$. We consider that $k \ll M'$ (for instance $k \approx 10$ as in [GFLL15]), thus $x < 2 \cdot M'$, the complexity of the decryption with Pollard's lambda algorithm is $O(f(2M')^{1/2}) \approx O(M'^{k/2})$. For all $1 \leq i \leq k$, the attacker computes $M'_i = k - i + M'$ and $y_i = y \mod M'_i$.

Since for all $a \in \mathcal{X}, M' > f(a)$, we have for all $1 \leq i \leq k, M'_i = k - i + M' \geq M' > f(a)$. Using Property 2 and since $i \in \mathcal{X}$, we deduce that

$$\begin{aligned} y_i &= f(x) \mod M'_i \\ &= f(k + M') \mod M'_i \\ &= f(k - i + i + M') \mod M'_i \\ &= f\left(i + M'_i\right) \mod M'_i = f(i) \ . \end{aligned}$$

Hence, the attacker obtains $k + 1$ points from one single queried point and uses Lagrange's interpolation on $((1, y_1), (2, y_2), \ldots, (k, y_k), (x, y))$ to guess f. Then, the attacker can compute f with reasonable computation time.

Now, we show the generalized case for any set $\mathcal{X}$ where $|\mathcal{X}| \geq k$. To begin, the attacker chooses a vector of k integers $(x_1, \ldots, x_k)$ such that, for all $1 \leq j \leq k, x_j > 0$ and: $x'_i = \left(\sum_{j=1}^{i} x_j\right)$ where $x'_i \in \mathcal{X}$. Then the attacker sends the query $x = \left(\sum_{i=1}^{k} x_i\right) + M'$ to the server such that $M' \in \mathbb{N}$ and for all $a \in \mathcal{X}$ we have $M' > f(a)$. After he sends the query to the server, the attacker receives the encryption of $y = f(x)$.

Pollard's lambda algorithm complexity [Pol78] on M' is $O(M'^{1/2})$. We consider that $k << M'$, $k \approx 10$ as in [GFLL15], thus $x < 2 \cdot M'$, the complexity of the decryption with Pollard's lambda algorithm is $O(f(2M')^{1/2}) \approx O(M'^{k/2})$.

With the $y = f(x)$ returned by the server, the attacker computes for all $1 \leq i \leq k$:

$$M'_i = \sum_{j=i+1}^{k} x_j + M' \ .$$

Then we define y_i for all $1 \leq i \leq k$ such that $y_i = y \mod M'_i$. Since for all $a \in \mathcal{X}, M' > f(a)$, we have for all $1 \leq i \leq k$ and for all $a \in \mathcal{X}$:

$$M'_i = \sum_{j=i+1}^{k} x_j + M' \geq M' > f(a) \ .$$

Using Properties 1 and 2 of Section 3 and since $x'_i \in \mathcal{X}$, we deduce:

$$\begin{aligned} y_i = f(x) \mod M'_i &= f\left(\sum_{i=1}^{k} x_i + M'\right) \mod M'_i \\ &= f\left(\sum_{j=i+1}^{k} x_j + \sum_{j=1}^{i} x_j + M'\right) \mod M'_i = f\left(\sum_{j=1}^{i} x_j + M'_i\right) \mod M'_i \\ &= f\left(\sum_{j=1}^{i} x_j\right) = f(x'_i) \ . \end{aligned}$$

Finally, the attacker knows the k points of f: $(x'_i, f(x'_i))$ for $1 \leq i \leq k$, and also $(x, f(x))$. Hence, using Lagrange's interpolation, the attacker is able to retrieve the polynomial f.

It is possible to attack the scheme of Gajera *et al.* [GND16] in a similar way. Indeed, as in [GFLL15], the user knows a value M such that $\forall x \in \mathcal{X}, f(x) < M$. A simple countermeasure could be to not allow the user to evaluate inputs that are not in $\mathcal{X}$. Unfortunally, this is not possible in these two schemes since the user encrypts his data x. Hence, the server does not know whether $x \in \mathcal{X}$ or not.

4 Security Models

We revisit the formal security models for PPE schemes for two main reasons: (i) Kate *et al.* [KZG10] propose some models where the secret polynomial is randomly chosen. However, they present several practical applications where the polynomial is not actually random, and some information, such as bounds for $f(x)$ or candidates for f, can be infered easily from the context. Their models are clearly not sufficient for analysing the security of this kind of applications. (ii) The schemes presented by Guo *et al.* [GFLL15] and Gajera *et al.* [GND16] consider polynomials that are not randomly chosen. The authors give neither security models nor security proofs. We show previously a practical attack on these two schemes where a user exploits some public information. To avoid such attacks, we need a model where public information does not give significant advantage.

Our goal is to design a model where the public parameters and the server's proofs of correctness give no advantage to an attacker. Ideally, we would like the attacker to have no more chances of guessing the polynomial than if he only had access to a server reliably returning polynomial evaluations with no proof of correctness. Our security model considers an attacker that tries to determine which polynomial is used by a PPE among two polynomials of his choice. This model is inspired by the IND-CPA model used in public key cryptography.

4.1 Formal Definition

In order to be able to define our security model, we first need to formally define a Private Polynomial Evaluation scheme.

Definition 8. *A* Private Polynomial Evaluation *(PPE) scheme is composed of four algorithms* $(\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *such that:*

$\mathsf{setup}(\lambda)$**:** *It returns a ring* F *and a public setup* pub*.*
$\mathsf{init}(\mathsf{pub}, f)$**:** *It returns a server key* sk *and a verification key* vk *according to the polynomial* $f \in F[X]$*.*
$\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$**:** *It returns* y *and a proof* π *that* $y = f(x)$*.*
$\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$**:** *It returns* 1 *if the proof* π *is "accepted" otherwise* 0*.*

4.2 Security models

We start be redifining the notion of weak security presented in the literature. We then introduce the notion of chosen function attack and the natural notion of unforgeability. Proofs for Theorems 2, 3 and 4 are given in [BDG+17].

Polynomial Protection We introduce the *Polynomial Protection* (PP) security. A PPE is PP-secure if no adversary can output a new point (not computed by the server) of the secret polynomial f with a better probability than by guessing. In this model, the polynomial is randomly chosen and the adversary cannot

use the server more than k times, where k is the degree of f. This security model is similar to the *Hiding Model* [KZG10] except that the adversary chooses the points to be evaluated. We define the *Weak Polynomial Protection* (WPP) as the same model as PP except that the adversary has no access to the server.

Definition 9 (PP and WPP). *Let Π be a PPE, $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary. $\forall k \in \mathbb{N}$, the* k-Polynomial Protection *(k-PP) experiment for $\mathcal{A}$ against Π denoted by $\mathsf{Exp}^{k\text{-}\mathsf{PP}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 3, where $\mathcal{A}$ has access to the* server oracle $\mathsf{CO}_{\mathsf{PP}}(\cdot)$. *We define the advantage of the adversary $\mathcal{A}$ against the k-PP experiment by:*

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

A scheme Π is k-PP-secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)$.

We define the k-Weak Polynomial Protection *(k-WPP) experiment as the k-PP experiment except that $\mathcal{A}$ does not have access to the oracle $\mathsf{CO}_{\mathsf{PP}}(\cdot)$. In a similar way, we define the WPP advantage and security.*

The only difference between PP and WPP is that the adversary has no access to the oracle in WPP, so PP security implies the WPP security.

Theorem 1. *For any Π and k, if Π is k-PP-secure then Π is k-WPP-secure.*

Chosen Function Attack We define a model for *indistinguishability against chosen function attack*. In this model, the adversary chooses two polynomials (f_0, f_1) and tries to guess the polynomial f_b used by the server, where $b \in \{0, 1\}$. The adversary has access to a server that evaluates and proves the correctness of $y = f_b(x)$ only if $f_0(x) = f_1(x)$. This is an inherent limitation: if the adversary can evaluate another point (x, y) such that $f_0(x) \neq f_1(x)$, then he can compare y with $f_0(x)$ and $f_1(x)$ and recover b. In practice, an adversary chooses (f_0, f_1) such that $f_0 \neq f_1$, but with k points (x_i, y_i) such that $f_0(x_i) = f_1(x_i)$. It allows the adversary to maximize his oracle calls in order to increase his chances of success. We remark that schemes [GFLL15] and [GND16] are not IND-CFA-secure: users know a value M and the set of inputs $\mathcal{X}$ such that $\forall x \in \mathcal{X}, f(x) < M$. An attacker may choose two polynomials f_0 and f_1 such that for a chosen a, $f_0(a) < M$ and $f_1(a) > M$. Since $\mathcal{X}$ is public, the attacker returns f_0 if and only if $a \in \mathcal{X}$.

Definition 10 (IND-CFA). *Let Π be a PPE, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a two-party PPT adversary and k be an integer. The* k-Indistinguishability against Chosen Function Attack *(k-IND-CFA) experiment for $\mathcal{A}$ against Π is defined in Figure 3, where $\mathcal{A}$ has access to the* server oracle $\mathsf{CO}_{\mathsf{CFA}}(\cdot)$. *The advantage of the adversary $\mathcal{A}$ against the k-IND-CFA experiment is given by:*

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

A scheme Π is k-IND-CFA-secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$.

$\mathbf{Exp}^{k\text{-}\mathsf{PP}}_{\Pi,\mathcal{A}}(\lambda)$:
$(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$;
$f \xleftarrow{\$} F[X]_k$;
$\Sigma \leftarrow \emptyset$;
$c \leftarrow 0$;
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$;
$(x_*, y_*) \leftarrow \mathcal{A}^{\mathsf{CO}_{\mathsf{PP}}(\cdot)}(\mathsf{pub}, \mathsf{vk}, F, k)$;
If $(x_*, y_*) \notin \Sigma$ and $f(x_*) = y_*$:
Then return 1;
Else return 0;

$\mathbf{Exp}^{\mathsf{UNF}}_{\Pi,\mathcal{A}}(\lambda)$:
$(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$;
$(f, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pub}, F)$;
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$;
$(x_*, y_*, \pi_*) \leftarrow \mathcal{A}_2(\mathsf{pub}, \mathsf{sk}, \mathsf{vk}, F, f, \mathsf{st})$;
If $f(x_*) \neq y_*$ and $\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x_*, y_*, \pi_*)$:
Then return 1;
Else return 0;

$\mathbf{Exp}^{k\text{-}\mathsf{IND\text{-}CFA}}_{\Pi,\mathcal{A}}(\lambda)$:
$b \xleftarrow{\$} \{0,1\}^*$;
$(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$;
$(f_0, f_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pub}, F, k)$;
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f_b)$;
$b_* \leftarrow \mathcal{A}_2^{\mathsf{CO}_{\mathsf{CFA}}(\cdot)}(\mathsf{pub}, \mathsf{vk}, F, k, \mathsf{st})$;
If $f_0 \notin F[X]_k$ or $f_1 \notin F[X]_k$:
Then return 0;
Else return $(b = b_*)$;

$\mathsf{CO}_{\mathsf{PP}}(x)$:
$(y, \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$;
$c \leftarrow c + 1$;
$\Sigma \leftarrow \Sigma \cup \{(x, y)\}$;
If $c = k + 1$:
Then return $\perp$;
Else return (y, π);

$\mathsf{CO}_{\mathsf{CFA}}(x)$:
$(y, \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f_b)$;
If $f_0(x) \neq f_1(x)$:
Then return $\perp$;
Else return (y, π);

Fig. 3: Security experiments and oracles definitions.

In Theorem 2, we prove that IND-CFA security implies WPP security: if there exists an adversary $\mathcal{A}$ against the WPP experiment who is able to *decrypt* a random polynomial from the public values, then we can use it to guess f_b in an IND-CFA experiment for any chosen polynomials (f_0, f_1). However, surprisingly, it is not true for the PP security (Theorem 3). The reason is that the oracle of the IND-CFA experiment has restriction, so it cannot be used to simulate the oracle of the PP experiment in a security reduction.

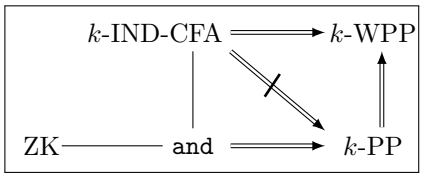

Fig. 4: Security relations.

Theorem 2. *If Π is a k-IND-CFA-secure PPE, then it is k-WPP-secure.*

Theorem 3. *Let Π be a k-IND-CFA-secure PPE, it does not imply that Π is k-PP.*

However, we would like to have a simple and sufficient condition under which the IND-CFA security implies the PP security. For this, we define the *proof induced*

by a PPE which is the proof algorithm used by the algorithm $\mathsf{compute}$. We show that if this proof system is zero-knowledge, then the IND-CFA security implies the PP security.

Definition 11. *Let $\Pi = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ be a PPE, the* non-interactive proof inducted by *Π, denoted $P_\Pi = (\mathsf{proof}_\Pi, \mathsf{ver}_\Pi)$ is defined as follows. For any $\lambda, k \in \mathbb{N}$, $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$, $f \in F[X]_k$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$:*

$\mathsf{proof}_\Pi((\mathsf{pub}, \mathsf{vk}, x, y), (f, \mathsf{sk}))$**:** *returns π, where $(y', \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$.*
$\mathsf{ver}_\Pi((\mathsf{pub}, \mathsf{vk}, x, y), \pi)$**:** *runs $b \leftarrow \mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$ and returns it.*

We say that Π is Zero-Knowledge *(ZK) if P_Π is Zero-Knowledge.*

Theorem 4. *Let Π be a ZK and k-IND-CFA-secure PPE, then Π is k-PP-secure.*

In Figure 4, we recall all relations between our security properties.

Unforgeability Finally, we define the unforgeability property for a PPE. A PPE is unforgeable when a dishonest server cannot produce a valid proof on the point (x, y) when $f(x) \neq y$. The secret polynomial f is chosen by the server.

Definition 12. *Let Π be a PPE, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a two-party PPT adversary. The* Unforgeability *(UNF) experiment for $\mathcal{A}$ against Π is defined in Figure 3. We define the advantage of the adversary $\mathcal{A}$ against the UNF experiment by:*

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

A scheme Π is UNF-secure if this advantage is negligible for any $\mathcal{A} \in \mathrm{POLY}(\lambda)^2$.

4.3 Security against collusion attacks

To conclude, our security model implicitly prevents all non-inherent collusion attacks, because in our context the clients have no secret information. There are two kinds of collusion scenarios:

A client colludes with the server: If a client colludes with the server, then the server can obviously give him the secret polynomial. This limitation is inherent and cannot be prevented. On the other hand, all keys known by the clients are public and known to the server, the server has no advantage in colluding with a client. In particular, the collusion does not allow the server to forge fake validity proofs for others clients.

Several clients collude together: All clients have the same verification keys. Thus, a client gains no advantage by colluding with other clients, as long as the total number of known points is less than k after collusion. Obviously the inherent limitation of PPE still holds: if the collusion of clients learn more than k points, then they can guess the polynomial.

5 PIPE Description

We recall Feldman's Verifiable Secret Sharing (VSS) scheme and build a simple k-PP PPE that is not k-IND-CFA. We then propose some modifications based on the Feldman's VSS and the ElGamal scheme in order design our secure PPE scheme PIPE that is k-IND-CFA. We analyse its security and compare it with the scheme of Kate *et al.* [KZG10].

5.1 Feldman's Verifiable Secret Sharing

Feldman's VSS [Fel87] is based on Shamir's Secret Sharing [Sha79], where each share is a point (x, y) of a secret polynomial f of degree k. Knowing more than k shares, one can guess the polynomial f and can compute the secret $s = f(0)$. In Feldman's VSS, there is a public value that allows anybody to check the validity of a share. For any point (x, y), anybody can check if y is $f(x)$ or not. This scheme works as follows. Let G be a multiplicative group of prime order p where DL is hard. Let $f \in \mathbb{Z}_p^*[X]$ be the secret polynomial and $a_i \in F$ be a coefficient for all $0 \leq i \leq k$ such that

$$f(x) = \sum_{i=0}^{k} a_i \cdot x^i \ .$$

Let $g \in G$ be a generator of G. For all $i \in \{0, \ldots, k\}$, we set $h_i = g^{a_i}$. Values g and $\{h_i\}_{0 \leq i \leq k}$ are public, however, the coefficients a_i are hidden under DL hypothesis. We remark that $f(x) = y$ if and only if $g^y = \prod_{i=0}^{k} h_i^{x^i}$ since

$$\prod_{i=0}^{k} h_i^{x^i} = \prod_{i=0}^{k} g^{a_i \cdot x^i} = g^{\sum_{i=0}^{k} a_i \cdot x^i} = g^{f(x)} \ .$$

Then, we can use it to check that (x, y) is a valid share.

5.2 Our Scheme: PIPE

Feldman's VSS can be used to design a PPE that is k-PP-secure: using the public values g and $\{h_i\}_{0 \leq i \leq k}$, any user can check that the point (x, y) computed by the server is a point of f. However, in a practical use, the polynomial f is not randomly chosen in a large set. An IND-CFA attacker knows that $f = f_0$ or $f = f_1$ for two known polynomials (f_0, f_1), since he knows the coefficients $\{a_{0,i}\}_{0 \leq i \leq k}$ and $\{a_{1,i}\}_{0 \leq i \leq k}$ of these two polynomials, he can compute the values $\{g^{a_{0,i}}\}_{0 \leq i \leq k}$ and $\{g^{a_{1,i}}\}_{0 \leq i \leq k}$ and he can compare it with the public set $\{h_i\}_{0 \leq i \leq k}$.

In order to construct our k-IND-CFA PPE, called PIPE, we give an ElGamal key pair $(\mathsf{pk}, \mathsf{sk})$ to the server where $\mathsf{pk} = (G, p, g, h)$ and $h = g^{\mathsf{sk}}$ and we encrypt all the h_i. Then for all $i \in \{0, \ldots, k\}$, the users do not know $h_i = g^{a_i}$ but know the ElGamal ciphertext (c_i, d_i) such that $c_i = g^{r_i}$ and $d_i = h^{r_i} \cdot h_i$ where r_i is randomly chosen. Since ElGamal is IND-CPA-secure, an attacker that chooses

two polynomials (f_0, f_1) cannot distinguish, for $0 \leq i \leq k$, if the ciphertext (c_i, d_i) encrypts a coefficient of f_0 or of f_1. Thus, the attacks on the previous scheme are no longer possible.

Moreover, the user can check that $f(x) = y$ for a point (x, y) using the values $\{(c_i, d_i)\}_{0 \leq i \leq k}$. We set $r(x) = \sum_{i=0}^{k} r_i \cdot x^i$. The user computes:

$$c = \prod_{i=0}^{k} c_i^{x^i} = \prod_{i=0}^{k} g^{r_i \cdot x^i} = g^{\sum_{i=0}^{k} r_i \cdot x^i} = g^{r(x)} \ .$$

On the other hand, he computes:

$$d' = \prod_{i=0}^{k} d_i^{x^i} = \left(\prod_{i=0}^{k} h^{r_i \cdot x^i} \right) \cdot \left(\prod_{i=0}^{k} g^{a_i \cdot x^i} \right) = h^{\sum_{i=0}^{k} r_i \cdot x^i} \cdot g^{\sum_{i=0}^{k} a_i \cdot x^i} = h^{r(x)} \cdot g^{f(x)} \ .$$

Finally, $(c, d') = (g^{r(x)}, h^{r(x)} \cdot g^{f(x)})$ is an ElGamal ciphertext of $g^{f(x)}$. Then, to convince the user that (x, y) is a valid point of f, the server proves that (c, d') is a ciphertext of g^y using a NIZKP of $\log_g(c) = \log_h(d'/g^y)$.

This leads us to the following formal definition of our scheme PIPE.

Definition 13. *Let* $\mathsf{PIPE} = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *be a PPE defined by:*

$\mathsf{setup}(\lambda)$**:** *Using the security parameter* λ*, it generates* G *a group of prime order* p *and a generator* $g \in G$*. It chooses a hash function* $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p^*$ *and it sets* $F = \mathbb{Z}_p^*$*. It sets* $\mathsf{pub} = (G, p, g, H)$ *and returns* (pub, F)*.*

$\mathsf{init}(\mathsf{pub}, f)$**:** *We set* $f(x) = \sum_{i=0}^{k} a_i \cdot x^i$*. This algorithm picks* $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p^*$ *and computes* $\mathsf{pk} = g^{\mathsf{sk}}$*. For all* $i \in \{0, \ldots, k\}$*, it picks* $r_i \xleftarrow{\$} \mathbb{Z}_p^*$ *and computes* $c_i = g^{r_i}$ *and* $d_i = \mathsf{pk}^{r_i} \cdot g^{a_i}$*. Finally, it sets* $\mathsf{vk} = (\{(c_i, d_i)\}_{0 \leq i \leq k}, \mathsf{pk})$ *and returns* $(\mathsf{vk}, \mathsf{sk})$*.*

$\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$**:** *Using* vk *which is equal to* $(\{(c_i, d_i)\}_{0 \leq i \leq k}, \mathsf{pk})$*, this algorithm picks* $\theta \xleftarrow{\$} \mathbb{Z}_p^*$ *and computes*

$$c = \prod_{i=0}^{k} c_i^{x^i} \ , \quad \pi = (g^\theta, c^\theta, \theta + H(g^\theta, c^\theta) \cdot \mathsf{sk}) \ .$$

Finally, it returns $(f(x), \pi)$*.*

$\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$**:** *Using* $\mathsf{vk} = (\{(c_i, d_i)\}_{0 \leq i \leq k}, \mathsf{pk})$ *and* $\pi = (A, B, \omega)$*, this algorithm computes*

$$c = \prod_{i=0}^{k} c_i^{x^i} \ , \quad d = \frac{\left(\prod_{i=0}^{k} d_i^{x^i} \right)}{g^y} \ .$$

If $g^\omega = A \cdot \mathsf{pk}^{H(A,B)}$ *and* $c^\omega = B \cdot d^{H(A,B)}$*, then the algorithm returns* 1*, else it returns* 0*.*

Table 1: Comparison of PIPE and $\mathsf{PolyCommit}_{\mathsf{Ped}}$.

	Setup size	Key size	Verif. cost	Pairing	Assumption	Security
PIPE	$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$	Paring free	DDH	IND-CFA
$\mathsf{PolyCommit}_{\mathsf{Ped}}$ [KZG10]	$\mathcal{O}(k)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Pairing based	t-SDH	IND-CFA

5.3 Security

We prove the security of PIPE in our security model:

Lemma 1. *For any $k \in \mathbb{N}$,* PIPE *is k-IND-CFA-secure under the DDH assumption in the ROM.*

Lemma 2. PIPE *is unconditionally ZK-secure in the ROM.*

Lemma 3. PIPE *is unconditionally UNF-secure in the ROM.*

Proofs of Lemmas 1 2 and 3 are presented in [BDG+17]. Using Lemma 4, Lemma 2 and Theorem 4, we have that PIPE is k-PP-secure. Hence, using Lemma 1 and Theorem 2, we deduce that PIPE is k-WPP-secure. Finally, we have the following theorem.

Theorem 5. *For any $k \in \mathbb{N}$,* PIPE *is is ZK, k-IND-CFA, k-PP, k-WPP and UNF-secure under the DDH assumption in the ROM.*

5.4 Comparison with $\mathsf{PolyCommit}_{\mathsf{Ped}}$

Kate *et al.* [KZG10] propose two CTP schemes that can be used as PPE schemes. Even if Kate *et al.* security model does not take into account IND-CFA security, we prove in [BDG+17] that one of these two schemes, called $\mathsf{PolyCommit}_{\mathsf{Ped}}$, is IND-CFA-secure. We recall the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme in Appendix A and we compare PIPE with this scheme in this section. Table 1 resumes this comparison.

The PIPE verification algorithm is in $\mathcal{O}(k)$ and the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ one is in constant time. However, the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ verification algorithm requires several pairing computations which are significantly costly in terms of computation time whereas PIPE only requires exponentiations and multiplication in a prime order group. Consequently, PIPE will be more efficient than $\mathsf{PolyCommit}_{\mathsf{Ped}}$ for sufficiently small polynomial degree k.

The main advantage of $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is that the verification key size is constant whereas the verification key size of PIPE is in $\mathcal{O}(k)$. However, the public setup size of $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is in $\mathcal{O}(k)$ whereas the PIPE one is in constant. Since the client knows both the verification key and the public setup, $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is advantageous only if each client has access to several polynomials simultaneously.

PIPE is secure under the DDH assumption whereas $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is secure under the t-SDH assumption. Note that finding a scheme that is secure under a weaker assumption than t-SDH was an open problem mentionned by Kate *et al.* [KZG10]. Finally, note that the security $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is proven in the

standard model. A simple way to obtain a version of PIPE that is secure in the standard model is to use the interactive version of LogEq [CP93] instead of the non-interactive one in the algorithm. In return, it requires an interaction between the client and the server during the evaluation algorithm.

6 CFA Security for Commitments to Polynomials

Our scheme can be used as a *commitment to polynomials* scheme [KZG10] that is CFA-secure. We give an overview of a such scheme in Figure 5. To commit a polynomial f, the committer computes $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$ and returns the commitment vk to the user corresponding to the encryption of coefficients of the polynomial f. Then, the user sends his data to the committer (x_i in Figure 5) and receives the results with correctness proof ($(f(x_i), \mathsf{proof})$ in Figure 5). To open the commitment, the committer reveals to the user the key vk together with f ($\mathsf{open}(\mathsf{vk}, f)$ in Figure 5), then the user can open all the ElGamal ciphertexts of vk and check that they encrypt g^{a_i} where a_i are the coefficients of f.

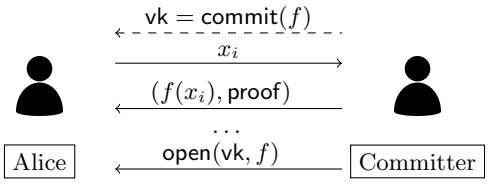

Fig. 5: PIPE scheme used as a commitment to polynomials scheme [KZG10].

7 Anonymous Private Polynomial Evaluation

In a practical scenario, the company does not allow anybody to interact freely with the computation server. The company distributes authentication keys to the clients, and the server uses a protocol to authenticate the client at the beginning of each interaction. It allows the server to verify that a client does not request to evaluate more than k points, where k is the degree of the polynomial. However, for a lot of applications, preserving the privacy of the clients is important. Guo *et al.* [GFLL15] propose an anonymous authentication mechanism for their scheme, which is broken and fixed by Gajera *et al.* [GND16].

We remark that anonymous authentication for PPE prevents the server from knowing how much points of the polynomial it gives to each client, leading to security issues. To solve this problem, we suggest that the server uses *k-times anonymous authentication* [TFS04]: this primitive allows a client to anonymously authenticate k times. If a client exceeds this limit, the server can identify him. Using such a scheme, the server can refuse to respond if the user requires more point evaluations than allowed, and the privacy of honest users is preserved.

8 Conclusion

In this paper, we gave a formal definition for a primitive called PPE. This primitive allows a company to delegate computations on a secret polynomial for users in a verifiable way. In essence, the user sends x and receives y from the server along with a proof of $y = f(x)$; even though he does not know the polynomial f. We proposed a security model of indistinguishability against chosen function attack (IND-CFA) and we built a PPE scheme called PIPE which is secure in this model. We proved that another scheme called $\mathsf{PolyCommit}_{\mathsf{Ped}}$ [KZG10] is IND-CFA-secure, and we compared it with PIPE. Moreover, we exhibited a critical flaw in two papers which proposed schemes tackling the same problem. In the future, we aim at designing a scheme that is pairing free and that uses constant size verification keys. Another possible extension is to add practical privacy mechanism to protect the data sent by the users.

Acknowledgements

This research was conducted with the support of the FEDER program of 2014-2020, the region council of Auvergne-Rhône-Alpes, the support of the "Digital Trust" Chair from the University of Auvergne Foundation, 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

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT 2000*. Springer, 2000.

[BDG+17] Xavier Bultel, Manik Lal Das, Hardik Gajera, David Grault, Matthieu Giraud, and Pascal Lafourcade. Verifiable private polynomial evaluation. Cryptology ePrint Archive, Report 2017/756, 2017. http://eprint.iacr.org/2017/756.

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, 1998. Invited paper.

[CKKC13] S. G. Choi, J. Katz, R. Kumaresan, and C. Cid. Multi-client non-interactive verifiable computation. In *TCC 2013*. Springer, 2013.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, August 1993.

[CRR12] Ran Canetti, Ben Riva, and Guy N. Rothblum. Two protocols for delegation of computation. In Adam Smith, editor, *ICITS 12*, volume 7412 of *LNCS*, pages 37–61. Springer, August 2012.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437. IEEE Computer Society Press, October 1987.

[FG12] D. Fiore and R. Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *ACM CCS 12*. ACM Press, 2012.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. *Keyword Search and Oblivious Pseudorandom Functions*, pages 303–324. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. *Efficient Private Matching and Set Intersection*, pages 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO86*. Springer, 1987.

[GFLL15] Linke Guo, Yuguang Fang, Ming Li, and Pan Li. Verifiable privacy-preserving monitoring for cloud-assisted mhealth systems. In *INFOCOM*. IEEE, 2015.

[GGP10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO 2010*. Springer, 2010.

[GND16] Hardik Gajera, Shruti Naik, and Manik Lal Das. On the security of "verifiable privacy-preserving monitoring for cloud-assisted mhealth systems". In *ICISS*. Springer, 2016.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT 2010*. Springer, 2010.

[LP02] Lindell and Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 245–254, New York, NY, USA, 1999. ACM.

[PHGR13] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013.

[Pol78] J.M. Pollard. A monte carlo method for index computation (mod p). In *Mathematics of Computation*, volume 32, pages 918–924. Springer, 1978.

[PRV12] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC 2012*. Springer, 2012.

[PST13] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In *TCC 2013*. Springer, 2013.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-times anonymous authentication (extended abstract). ASIACRYPT 2004, pages 308–322. Springer, 2004.

A $\mathsf{PolyCommit}_{\mathsf{Ped}}$ Scheme [KZG10]

We recall the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ construction presented by Kate *et al.* [KZG10].

Definition 14. $\mathsf{PolyCommit}_{\mathsf{Ped}} = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *is a* PPE *scheme defined as follows:*

$\mathsf{setup}(\lambda)$**:** *Using the security parameter* λ*, it generates two groups* $\mathbb{G}$ *and* $\mathbb{G}_T$ *of prime order* p *(providing* λ*-bit security) such that there exists a symmetric bilinear pairing* $e\colon \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$*. Moreover, it chooses two generators* g *and* h *of* $\mathbb{G}$ *and picks* $\alpha \leftarrow \mathbb{Z}_p^*$*. It sets* $F = \mathbb{Z}_p^*$*,* $\mathsf{pub} = (\mathbb{G}, \mathbb{G}_T, p, e, g, h, (g^{\alpha}, \ldots, g^{\alpha^k}), (h^{\alpha}, \ldots, h^{\alpha^k}))$ *and returns* (pub, F)*.*

$\mathsf{init}(\mathsf{pub}, f)$**:** *Using* $f(x) = \sum_{i=0}^{k} a_i \cdot x^i$*, this algorithm chooses a random polynomial of degree* k*,* $r(x) = \sum_{i=0}^{k} r_i \cdot x^i \in \mathbb{Z}_p[x]$ *and sets* $\mathsf{sk} = r(x)$*. It computes* $\mathcal{C} = \prod_{i=0}^{k} (g^{\alpha^i})^{a_i} (h^{\alpha^i})^{r_i} = g^{f(\alpha)} h^{r(\alpha)}$ *and sets* $\mathsf{vk} = \mathcal{C}$*. Finally, it returns* $(\mathsf{sk}, \mathsf{vk})$*.*

$\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x_i, \mathsf{sk}, f)$**:** *This algorithm computes* $\psi_i(x) = (f(x) - f(x_i))/(x - x_i)$ *and* $\hat{\psi}_i(x) = (r(x) - r(x_i))/(x - x_i)$*. Let* $(\gamma_0, \ldots, \gamma_k)$ *and* $(\hat{\gamma}_0, \ldots, \hat{\gamma}_k)$ *be such that* $\psi_i(x) = \sum_{j=0}^{k} \gamma_j \cdot x^j$ *and* $\hat{\psi}_i(x) = \sum_{j=0}^{k} \hat{\gamma}_j \cdot x^j$*. It computes* $w_i = \prod_{j=0}^{k} (g^{\alpha^j})^{\gamma_j} (h^{\alpha^j})^{\hat{\gamma}_j} = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$*. It sets* $\pi = (x_i, r(x_i), w_i)$ *and returns* $(f(x_i), \pi)$*.*

$\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x_i, f(x_i), \pi)$**:** *If* $e(\mathcal{C}, g)$ *equals to* $e(w_i, (g^{\alpha})^{-x_i}) e(g^{f(x_i)} h^{r(x_i)}, g)$*, the algorithm outputs* 1*, else it outputs* 0*.*