Practically Efficient Verifiable Delegation of Polynomial and its Applications

Jia Xu

National University of Singapore
Department of Computer Science
jiaxu2001@gmail.com

Abstract. In this paper, we propose a novel one-way function, which is equivalent to large integer factorization. From this new one-way function, we construct a novel verifiable delegation scheme for polynomial. Our contribution is twofold: in the practice aspect, our proposed polynomial delegation scheme is about 100 times faster than the existing solutions [1, 2] and has constant key size where the existing works require linear key size w.r.t. the degree of the delegated polynomial; in the theory part, our proposed scheme is provably secure under large integer factorization, which is a much weaker assumption than that of existing works. The efficient polynomial delegation scheme can be applied in constructing proofs of retrievability scheme, verifiable keyword search and verifiable dictionary data structure and so on. Furthermore, our new one-way function may have independent interests.

Keywords: Cloud Computing, Verifiable Remote Computing, Delegation of Polynomial, Proofs of Storage, One-way function

1 Introduction

Verifiable remote computing is an important research topic in secure cloud computing. Leveraging on Gentry's fully homomorphic encryption scheme [3], Gennaro *et al.* [4] and Chung *et al.* [5] gave two solutions to delegate any polynomial time computable function in the verifiable remote computing model. Although these two generic solutions are asymptotically efficient, researchers are still pursuing practically efficient delegation schemes, even for a small class of functions. Recently, Kate *et al.* [1] proposed a polynomial commitment scheme with constant proof size. Benabbas *et al.* [2] proposed a verifiable remote computing protocol for polynomials. Both schemes are provably secure under (computational or decisional) Strong Diffie-Hellman Assumption, and requires at least linear key size and linear number of exponentiation operations w.r.t. the degree of the polynomial.

In this paper, we devise a novel one-way function that is equivalent to the large integer factorization. We manage to replace the Strong Diffie-Hellman Assumption with our new one-way function, and construct a new delegation scheme for polynomial. Compared with the existing works, our proposed scheme has the following improvements:

- The number of exponentiation operations is reduced from $\mathcal{O}(d)$ to $\mathcal{O}(1)$, where d is the degree of the polynomial.
- The key size reduce from $\mathcal{O}(d)$ to $\mathcal{O}(1)$.
- Our scheme relies on large integer factorization, which is a much weaker assumption compared with Strong Diffie-Hellman Assumption.

The details of comparison between our scheme and existing works are in Table 1.

Table 1: Comparison of the proposed scheme with state of arts. Suppose ℓ number of polynomial of degree d are delegated.

Scheme	Key Size	Storage overhead	Computation (Preprocess)	Computation (Prover)	Computation (Verifier)
PolyCommit [1]	$\mathcal{O}(d)$	$\mathcal{O}(\ell)$	$d\ell$ `exp`	d `exp`	1 `pairing`
PolyDelegation [2]	$\mathcal{O}(d)$	$\mathcal{O}(d\ell)$	$d\ell$ `exp`	d `exp`	2 `exp`
This paper	$\mathcal{O}(1)$	$\mathcal{O}(\ell)$	$d\ell$ `mul` + ℓ `PRF`	6 `exp` + $\mathcal{O}(d)$ `mul`	2 `exp`

2 New One-Way Functions based on Large Integer Factorization

In this section, we construct several new one-way functions, based on large integer factorization problem.

Let $n = pq$ be a RSA modulus, where both p and q are safe primes and the bit lengths of p and q are close. Let α, β be two secret numbers chosen from $\mathbb{Z}_n^*$ at random. For each integer i, we define g_i as below

$$g_i \stackrel{\text{def}}{=} \alpha^i + \beta^i \mod n.$$

It is well known that finding square root modulo a RSA modulus n is equivalent to factorizing n [6].

2.1 The first one-way function F_1

$$\mathsf{F}_1(\alpha, \beta) \stackrel{\text{def}}{=} (g_1, g_2) = (\alpha + \beta, \alpha^2 + \beta^2) \pmod n. \tag{1}$$

Lemma 1 F_1 *is a (strong) one-way function if it is computationally hard to factorize n.*

2.2 The second one-way function F_2

Lemma 2 *Let us define a function G as below.*

$$\mathsf{G}(g_1, g_2, k, m) \stackrel{\text{def}}{=} (g_k, g_{k+1}, g_{k+2}, \ldots, g_{k+m-1}) \pmod n, \text{ where } k, m \in \mathbb{Z}_n. \tag{2}$$

There exits a deterministic algorithm with $\mathcal{O}(m + \log k)$ modular multiplications/additions to compute G.

Proof (of Lemma 4). We can compute g_k recursively. Let us define three functions f_1, f_2, f_3 as below.

$$f_1(g_k, \alpha^k\beta^k) = (g_{2k}, \alpha^{2k}\beta^{2k}) \pmod n \tag{3}$$

$$f_2(g_1, g_k, g_{k+1}, \alpha\beta) = (g_1, g_{k+1}, g_{k+2}, \alpha\beta) \pmod n \tag{4}$$

$$f_3(g_k, g_{k+1}, \alpha^k\beta^k) = (g_{2k}, g_{2k+1}, \alpha^{2k}\beta^{2k}) \pmod n \tag{5}$$

It is easy to verify that all functions f_1 and f_2 and f_3 can be computed in $\mathcal{O}(1)$ multiplications/additions. With function f_2 and f_3, one can compute $(g_k, g_{k+1}, g_{k+2}, \ldots, g_{k+m-1})$ in $\mathcal{O}(m + \log k)$ number of multiplications/additions.

Corollary 3 *The function F_2 defined as below is a (strong) one-way function, if it is computationally hard to factorize n.*

$$\mathsf{F}_2(\alpha, \beta, k, m) \stackrel{\text{def}}{=} (k, m, g_k, g_{k+1}, \ldots, g_{k+m-1}) \pmod n \tag{6}$$

Corollary 4 *For each integer $c \geq 1$, there is an efficient deterministic algorithm with complexity in $\mathcal{O}(m + \log k)$ to compute $\{g_{ck} : k \in [3, m+2]\}$ given g_c, g_{2c} as input.*

2.3 The third one-way function F_3

Let us define function F_3 as below

$$\mathsf{F}_3(g_1, g_2) \stackrel{\text{def}}{=} (g_2, g_4) \pmod n. \tag{7}$$

Lemma 5 F_3 *is a (strong) one-way function, if it is computationally hard to factorize n.*

2.4 The fourth one-way function

More generally, for each integer $c \geq 2$, we have $\mathsf{G}^{(c)}$

$$\mathsf{G}^{(c)}(g_1, g_2, k, m) = \{g_{ck} : c \geq 2, k \in [1, m]\}. \tag{8}$$

Lemma 6 $\mathsf{G}^{(c)}$ *is a (strong) one-way function, if it is computationally hard to factorize n.*

2.5 Sequence $\langle g_i \rangle$

For any $a \in \mathbb{Z}_n^*$, let $\mathsf{ord}_n(a)$ denote the multiplicative order of a modulo n, i.e. $\mathsf{ord}_n(a)$ is the smallest positive integer k such that

$$a^k = 1 \mod n.$$

Lemma 7 *Let set $\mathbb{G}_1 = \{g_i : i \in \mathbb{N}\}$ and set $\mathbb{G}_2 = \{(g_i, g_{i+1}) : i \in \mathbb{N}\}$. We have*

- *Both $\mathbb{G}_1$ and $\mathbb{G}_2$ are finite sets.*
- *The size of finite set $\mathbb{G}_2$ is $|\mathbb{G}_2| = lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta))$, where $lcm(\cdot,\cdot)$ denotes the* least common multiplier*.*
- *The size of finite set $\mathbb{G}_1$ satisfies: $|\mathbb{G}_1| \geq \sqrt{|\mathbb{G}_2|}$.*

Proof (of Lemma 7).

Part I of Proof of Lemma 7: First of all, the sizes of sets $\mathbb{G}_1$ and $\mathbb{G}_2$ are finite:

$$|\mathbb{G}_1| \leq lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta)); \quad |\mathbb{G}_2| \leq lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta)).$$

Part II of Proof of Lemma 7: For any $x \in \mathbb{N}$, $g_x = \alpha^x + \beta^x = \alpha^x \cdot \alpha^{lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta))} + \beta^x \cdot \beta^{lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta))} = g_{x+lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta))}$. Thus, if $y = x + lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta))$, then $(g_x, g_{x+1}) = (g_y, g_{y+1})$.

On the other hand, we want to show that for any two distinct elements $x, y \in \mathbb{Z}_n$, *if* $(g_x, g_{x+1}) = (g_y, g_{y+1})$, *then* $lcm(\mathsf{ord}_n(\alpha), \mathsf{ord}_n(\beta)) | (y - x)$.

Let $\Delta = \alpha^x - \alpha^y$.From $\alpha^x + \beta^x = g_x = g_y = \alpha^y + \beta^y$, we have $\beta^x = \beta^y - \Delta$.

$$g_{x+1} = \alpha^{x+1} + \beta^{x+1} = (\alpha^y + \Delta)\alpha + (\beta^y - \Delta)\beta = \alpha^{y+1} + \beta^{y+1} + \Delta(\alpha - \beta) = g_{y+1} + \Delta(\alpha - \beta). \tag{9}$$

Since $g_{x+1} = g_{y+1}$, we have $\Delta(\alpha - \beta) = 0 \mod n$. Hence, either $\Delta = 0$ or $\alpha = \beta$.

In the case that $\Delta = 0$: $\alpha^x = \alpha^y \Rightarrow \alpha^{y-x} = 1 \Rightarrow \mathsf{ord}_n(\alpha)|(y - x)$. For the similar reason, $\mathsf{ord}_n(\beta)|(y - x)$. Thus, $lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta))|(y - x)$.

In the case that $\alpha = \beta$: $2\alpha^x = g_x = g_y = 2\alpha^y \Rightarrow \alpha^{y-x} = 1 \Rightarrow \mathsf{ord}_n(\alpha)|(y - x)$. Thus, $lcm(\mathsf{ord}_n(\alpha),\ \mathsf{ord}_n(\beta)) = \mathsf{ord}_n(\alpha)|(y - x)$.

Part III of Proof of Lemma 7 Each element in $\mathbb{G}_2$ is a pair of elements in the set $\mathbb{G}_1$. The number of all distinct pairs of elements from a set $\mathbb{G}_1$ is bounded by $|\mathbb{G}_1|^2$. Therefore, we have

$$|\mathbb{G}_2| \leq |\mathbb{G}_1|^2.$$

□

Example 1 *If $\beta = -\alpha$, the sequence g_i is*

$$2, 0, 2\alpha^2, 0, 2\alpha^4, 0, 2\alpha^6, 0, 2\alpha^8, 0, \ldots,$$

3 Delegation of Polynomial

In this section, we construct a new verifiable delegation scheme for polynomial, by employing the newly constructed one-way functions. Our construction also utilizes an intriguing algebraic property of polynomial: *for any polynomial $f(x)$ and scalar input r, the polynomial $(x - r)$ divides the polynomial $f(x) - f(r)$.*

3.1 Construction

$\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$

Find a λ bits long RSA modulus $n = pq$, where both p and q are safe primes. Choose α, β, τ at random from $\mathbb{Z}_n^*$. Choose a random PRF key and denote the key as seed. Let $g_i := \alpha^i + \beta^i \mod n$ for each integer $i \geq 0$. The public key is $pk = (n, g_1, g_2)$ and the private key is $sk = (n, \alpha, \beta, \tau, \mathsf{seed})$.

$\mathsf{Setup}(sk, \vec{m}) \rightarrow (\mathsf{id}, \sigma)$

The input is the coefficient vector $\vec{m} = (m_0, m_1, \ldots, m_{d-1})$ of the polynomial that is to be delegated. Choose a unique identifier id from $\{0, 1\}^{160}$. Compute σ as below:

$$\sigma := \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}) + \tau f_{\vec{m}}(\alpha) \mod n \tag{10}$$

Output (id, σ).

$\langle \mathsf{Eval}(pk, \mathsf{id}, \vec{m}, \sigma), \mathsf{Verify}(sk, r) \rangle \rightarrow$ accept or reject

Round 1:

- The verifier: Set $r_0 := r$, and choose r_1 at random from $\mathbb{Z}_n^*$: $r_1 \xleftarrow{\$} \mathbb{Z}_n^*$, and sends r_0, r_1 to the prover.
- The prover: For each $\iota \in \{0, 1\}$, generate $(y_\iota, \psi_{\iota,0})$ as below. Compute $y_\iota := f_{\vec{m}}(r_\iota)$. Divide the polynomial $f_{\vec{m}}(x) - f_{\vec{m}}(r_\iota)$ with polynomial $x - r_\iota$ using polynomial long division, and denote the resulting quotient polynomial as $f_{\vec{w}_\iota}(x)$, where $\vec{w}_\iota = (w_{\iota,0}, w_{\iota,1}, \ldots, w_{\iota,d-1})$. Compute $\psi_{\iota,0}$

$$\psi_{\iota,0} := \sum_{j=0}^{d-1} w_{\iota,j} g_j = \sum_{j=0}^{d-1} w_{\iota,j} (\alpha^j + \beta^j) \mod n. \tag{11}$$

  Send σ and $\{(y_\iota, \psi_{\iota,0}) : \iota \in \{0, 1\}\}$ to the verifier.

Round 2:

- The verifier: Choose k from $\mathbb{Z}_{\phi(n)}$ at random: $k \xleftarrow{\$} \mathbb{Z}_{\phi(n)}$. Send k to the verifier.
- The prover: For each $\iota \in \{0,1\}$ and $\nu \in \{k, k+1\}$, compute $\psi_{\iota,\nu}$ as below

$$\psi_{\iota,\nu} := \sum_{j=0}^{d-1} w_{\iota,j} g_{j+\nu} = \sum_{j=0}^{d-1} w_{\iota,j}(\alpha^{j+\nu} + \beta^{j+\nu}) \mod n. \tag{12}$$

  Send $\{\psi_{\iota,\nu} : \iota \in \{0,1\}, \nu \in \{k, k+1\}\}$ to verifier.
- The verifier: For each $\iota \in \{0,1\}$ and each $\nu \in \{k, k+1\}$, verify whether the following equality holds:

$$\frac{\tau^{-1}\left(\sigma - \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id})\right) \;-\; y_\iota}{\alpha - r_\iota} \stackrel{?}{=} \frac{\psi_{\iota,0}\beta^\nu - \psi_{\iota,\nu}}{\beta^\nu - \alpha^\nu} \mod n \tag{13}$$

  If all verifications succeed, output accept; otherwise, output reject.

3.2 Security

The security can be proved under quadratic residue hard problem modulo n, which is equivalent to factorization of n.

3.2.1 Completeness

Lemma 8 *The above polynomial delegation scheme is complete with overwhelming high probability.*

Proof (of Lemma 8). Let $\iota \in \{0,1\}$ and $A_\iota = f_{\vec{w}_\iota}(\alpha)$ and $B_\iota = f_{\vec{w}_\iota}(\beta)$.

From Equation (11), we have

$$\psi_{\iota,0} = \sum_{j=0}^{d-1} w_{\iota,j} g_j = \sum_{j=0}^{d-1} w_{\iota,j}(\alpha^j + \beta^j) = \sum_{j=0}^{d-1} w_{\iota,j}\alpha^j \;+\; \sum_{j=0}^{d-1} w_{\iota,j}\beta^j = f_{\vec{w}_\iota}(\alpha) + f_{\vec{w}_\iota}(\beta) = A_\iota + B_\iota \mod n.$$

From Equation (12), we have

$$\begin{aligned}\psi_{\iota,\nu} &= \sum_{j=0}^{d-1} w_{\iota,j} g_{j+\nu} = \sum_{j=0}^{d-1} w_{\iota,j}(\alpha^{j+\nu} + \beta^{j+\nu}) = \alpha^\nu \sum_{j=0}^{d-1} w_{\iota,j}\alpha^j \;+\; \beta^\nu \sum_{j=0}^{d-1} w_{\iota,j}\beta^j \\ &= \alpha^\nu f_{\vec{w}_\iota}(\alpha) + \beta^\nu f_{\vec{w}_\iota}(\beta) = \alpha^\nu A_\iota + \beta^\nu B_\iota \mod n.\end{aligned}$$

As a result, we have the linear equation system

$$\begin{cases} A_\iota + B_\iota = \psi_{\iota,0} \mod n & (i \in \{0,1\}) \\ \alpha^\nu A_\iota + \beta^\nu B_\iota = \psi_{\iota,\nu} \mod n & (\nu \in \{k, k+1\}) \end{cases} \tag{14}$$

Solving the above linear system, we find

$$\begin{cases} A_\iota = \frac{\psi_{\iota,0}\beta^\nu - \psi_{\iota,\nu}}{\beta^\nu - \alpha^\nu} \mod n \\ B_\iota = \frac{\psi_{\iota,0}\alpha^\nu - \psi_{\iota,\nu}}{\alpha^\nu + \beta^\nu} \mod n \end{cases} \tag{15}$$

Note that $f_{\vec{w}_\iota}(x)$ satisfies

$$\frac{f_{\vec{m}}(x) - f_{\vec{m}}(r_\iota)}{x - r_\iota} = f_{\vec{w}_\iota}(x) \tag{16}$$

By substituting $x = \alpha$ and $y_\iota = f_{\vec{\boldsymbol{m}}}(r_\iota)$ and $f_{\vec{\boldsymbol{m}}}(\alpha) = \tau^{-1}(\sigma - \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id}))$ and $A_\iota = f_{\vec{\boldsymbol{w}}_\iota}(\alpha)$ into the above equation, we have

$$\frac{\tau^{-1}\left(\sigma - \mathsf{PRF}_{\mathsf{seed}}(\mathsf{id})\right) \; - \; y_\iota}{\alpha - r_\iota} \quad \equiv \quad \frac{\psi_{\iota,0}\beta^\nu - \psi_{\iota,\nu}}{\beta^\nu - \alpha^\nu} \mod n \tag{17}$$

□

3.2.2 Soundness

Theorem 9 *The proposed delegation of polynomial scheme is sound.*

Proof (of Theorem 9). Let $(\sigma, \{(y_\iota, \psi_{\iota,0}, \psi_{\iota,k}, \psi_{\iota,k+1}) : \iota \in \{0,1\}\})$ be the correct proof. Suppose the adversary forges a valid but not correct proof $(\hat{\sigma}, \{(\hat{y}_\iota, \hat{\psi}_{\iota,0}, \hat{\psi}_{\iota,k}, \hat{\psi}_{\iota,k+1}) : i \in \{0,1\}\})$.

If $\hat{\sigma} \neq \sigma$, then the adversary can find the value of τ, which is protected by the PRF. That is, such adversary can break the PRF.

If $\hat{\sigma} = \sigma$, then the adversary can find $\Delta_{y,\iota} = y_\iota - \hat{y}_\iota, \Delta_{\iota,0} = \psi_{\iota,0} - \hat{\psi}_{\iota,0}, \Delta_{\iota,\nu} = \psi_{\iota,\nu} - \hat{\psi}_{\iota,\nu}$, $\nu \in \{k, k+1\}$ such that

$$\frac{-\Delta_{y,\iota}}{\alpha - r_\iota} = \frac{\Delta_{\iota,0}\beta^\nu - \Delta_{\iota,\nu}}{\beta^\nu - \alpha^\nu} \mod n \tag{18}$$

We substitute $\beta^\nu = g_\nu - \alpha^\nu$ into the above equation and obtain

$$\Delta_{\iota,0}\alpha^{\nu+1} + \left(2\Delta_{y,\iota} - r_\iota\Delta_{\iota,0}\right)\alpha^\nu + \left(\Delta_{\iota,\nu} - \Delta_{\iota,0}g_\nu\right)\alpha = \Delta_{y,\iota}g_\nu + r_\iota\Delta_{\iota,\nu} - r_\iota\Delta_{\iota,0}g_\nu \mod n \tag{19}$$

By replacing $(\alpha^{k+2}, \alpha^{k+1}, \alpha^k, \alpha)$ with (x_1, x_2, x_3, x_4) respectively, we obtain a a linear system in field $\mathbb{Z}_n^*$ in unknowns (x_1, x_2, x_3, x_4) as below:

$$\begin{bmatrix} 0 & \Delta_{0,0} & (2\Delta_{y,0} - r_0\Delta_{0,0}) & (\Delta_{0,k} - \Delta_{0,0}g_k) \\ 0 & \Delta_{1,0} & (2\Delta_{y,1} - r_1\Delta_{1,0}) & (\Delta_{1,k} - \Delta_{1,0}g_k) \\ \Delta_{0,0} & (2\Delta_{y,0} - r_0\Delta_{0,0}) & 0 & (\Delta_{0,k+1} - \Delta_{0,0}g_{k+1}) \\ \Delta_{1,0} & (2\Delta_{y,1} - r_1\Delta_{1,0}) & 0 & (\Delta_{1,k+1} - \Delta_{1,0}g_{k+1}) \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \Delta_{y,0}g_k + r_0\Delta_{0,k} - r_0\Delta_{0,0}g_k \\ \Delta_{y,1}g_k + r_1\Delta_{1,k} - r_1\Delta_{1,0}g_k \\ \Delta_{y,0}g_{k+1} + r_0\Delta_{0,k+1} - r_0\Delta_{0,0}g_{k+1} \\ \Delta_{y,1}g_{k+1} + r_1\Delta_{1,k+1} - r_1\Delta_{1,0}g_{k+1} \end{bmatrix} \tag{20}$$

To simplify the notations, we write the above linear system as below

$$\begin{bmatrix} 0 & M_{0,1} & M_{0,2} & M_{0,3} \\ 0 & M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,0} & M_{2,1} & 0 & M_{2,3} \\ M_{3,0} & M_{3,1} & 0 & M_{3,3} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix} \tag{21}$$

Our goal is to prove that: *if adversary wins, that is* $(\Delta_{y,0}, \Delta_{y,1}, \Delta_{0,0}, \Delta_{1,0}, \Delta_{0,k}, \Delta_{1,k}, \Delta_{0,k+1}, \Delta_{1,k+1}) \neq \vec{\mathbf{0}}$*, then the above linear system has a unique solution.* In that case, the adversary can solve the linear system to find α—Contradiction!

Since the above linear system has at least one solution $(x_1, x_2, x_3, x_4) = (\alpha^{k+1}, \alpha^{k+1}, \alpha^k, \alpha)$, the linear system has either unique solution or many (about $\phi(n)$) solutions.

We assume the linear system has many solutions. By Cramer's Rule, five determinants have to equal to 0:

$$
\begin{aligned}
0 &= \begin{vmatrix} 0 & M_{0,1} & M_{0,2} & M_{0,3} \\ 0 & M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,0} & M_{2,1} & 0 & M_{2,3} \\ M_{3,0} & M_{3,1} & 0 & M_{3,3} \end{vmatrix} = \begin{vmatrix} D_0 & M_{0,1} & M_{0,2} & M_{0,3} \\ D_1 & M_{1,1} & M_{1,2} & M_{1,3} \\ D_2 & M_{2,1} & 0 & M_{2,3} \\ D_3 & M_{3,1} & 0 & M_{3,3} \end{vmatrix} = \begin{vmatrix} 0 & D_0 & M_{0,2} & M_{0,3} \\ 0 & D_1 & M_{1,2} & M_{1,3} \\ M_{2,0} & D_2 & 0 & M_{2,3} \\ M_{3,0} & D_3 & 0 & M_{3,3} \end{vmatrix} \\
&= \begin{vmatrix} 0 & M_{0,1} & D_0 & M_{0,3} \\ 0 & M_{1,1} & D_1 & M_{1,3} \\ M_{2,0} & M_{2,1} & D_2 & M_{2,3} \\ M_{3,0} & M_{3,1} & D_3 & M_{3,3} \end{vmatrix} = \begin{vmatrix} 0 & M_{0,1} & M_{0,2} & D_0 \\ 0 & M_{1,1} & M_{1,2} & D_1 \\ M_{2,0} & M_{2,1} & 0 & D_2 \\ M_{3,0} & M_{3,1} & 0 & D_3 \end{vmatrix}
\end{aligned} \tag{22}
$$

In the Round 2, all variables are fixed except g_k, g_{k+1} and $\Delta_{\iota,\nu}, i \in \{0,1\}, \nu \in \{k, k+1\}$. Upon receiving a random k from the verifier, the adversary can compute the values of g_k, g_{k+1}. Next, the adversary has to find four values $\Delta_{\iota,\nu}, i \in \{0,1\}, \nu \in \{k, k+1\}$, such that the above five determinants equal to 0 (*Note this is the necessary but insufficient condition to allow the above linear system has many solutions*), where these determinants are determined by the value of $(g_k, g_{k+1}) \in \mathbb{G}_2$. That is, the adversary has to solve an equation system in four unknowns with five equations, of which three are quadratic equations and two are linear equations. Since $|\mathbb{G}_2| = lcm(\mathsf{ord}_n(\alpha), \mathsf{ord}_n(\beta))$ is exponentially large and $|\mathbb{G}_1| \geq \sqrt{|\mathbb{G}_2|}$, the chance that such solution exists is negligible.

4 Homomorphism and Two-Variable Polynomial

The proposed polynomial delegation scheme is homomorphic and this homomorphic property will lead a solution for delegating two-variable polynomial and an efficient proofs of retrievability scheme.

Suppose ℓ polynomials $f_{\vec{\boldsymbol{m}}_i}$, $i \in [\ell]$, are delegated, and denote with $(\mathsf{id}_i, \sigma_i)$ the output of Setup on polynomial $f_{\vec{\boldsymbol{m}}_i}$.

4.1 Homomorphism

HomEval

Receive input $(\vec{\boldsymbol{c}}, r)$ from the verifier. Parse the received vector $\vec{\boldsymbol{c}}$ as $(c_0, c_1, \ldots, c_{\ell-1})$. Compute $\vec{\boldsymbol{m}} := \sum_{i=0}^{\ell-1} c_i \vec{\boldsymbol{m}}_i$ and $\sigma := \sum_{i=0}^{\ell-1} c_i \sigma_i$. Then treat σ as the authentication tag of the polynomial $f_{\vec{\boldsymbol{m}}}$ and run the polynomial delegation scheme to evaluate $f_{\vec{\boldsymbol{m}}}(r)$.

4.2 Two-Variable Polynomial

A two variable polynomial can be written as

$$
f(x, y) = (1, y, y^2, \ldots, y^{\ell-1}) \times \begin{bmatrix} \vec{\boldsymbol{m}}_0 \\ \vec{\boldsymbol{m}}_1 \\ \vdots \\ \vec{\boldsymbol{m}}_{\ell-1} \end{bmatrix} \times (1, x, x^2, \ldots, x^d)^\top
$$

Delegate uni-variable polynomial $f_{\vec{\boldsymbol{m}}_i}$, $i \in [\ell - 1]$. To evaluate the two-variable polynomial $f(x, y)$ at point (x_0, y_0) Run the algorithm $\mathsf{HomEval}$ with coefficient $\vec{\boldsymbol{c}} = (1, y_0, y_0^2, \ldots, y_0^{\ell-1})$ and $r = x_0$.

5 Privacy Preserving

We can encrypt each coefficient $m_{i,j}$ of the delegated polynomial as $m_{i,j} + \mathsf{PRF}(id, i, j) \mod n$. The homomorphism of this encryption method ensures that the evaluation result can be extracted from the server's authenticated response.

6 Conclusion

In this paper, we proposed a new one-way function based on large integer factorization. From this one-way function, we constructed a new verifiable delegation scheme for polynomial, which improves the existing works in both practice and theory aspects.

References

1. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: ASIACRYPT. (2010) 177–194
2. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable Delegation of Computation over Large Datasets. In: CRYPTO. Volume 6841. (2011) 110
3. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: STOC '09: ACM symposium on Theory of computing. (2009) 169–178
4. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology. (2010) 465–482
5. Chung, K.M., Kalai, Y., Vadhan, S.P.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology. (2010) 483–501
6. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. Technical report, Cambridge, MA, USA (1979)