Privacy-preserving Fair Outsourcing Polynomial Computation without FHE and FPR

Ying Wang[1], Qiang Wang[1*], Zhifan Huang[1], Fucai Zhou[1], Changsheng Zhang[1], and Che Bian[2]

[1] Software College, Northeastern University, China
624718731@qq.com, wangqiang1@mail.neu.edu.cn, zeefinehy@163.com, {fczhou, zhangchangsheng}@mail.neu.edu.cn
[2] The Fourth Affiliated Hospital, China Medical University,China.
bianche1028@126.com

Abstract

Due to the rapid development of cloud computing, outsourcing computation has received considerable attention in recent years. Particularly, many outsourcing computation schemes have been proposed to dedicate the outsourcing polynomial computation due to its use in numerous fields, such as data analysis and machine learning. However, none of these schemes are practical enough because they either do not consider privacy or support the public verifiably to ensure fairness. To solve these problems, this paper proposes a new outsourced polynomial computation scheme combining paillier encryption and blockchain technology. Our scheme not only ensures the privacy of user data but also supports public verification, ensuring fairness between users and cloud servers. To achieve public verifiably, we apply the SGX technique, which is efficient in our proposal. Additionally, we implemented a prototype of our proposal and ran it on an Ethereum test net. Extensive experimental results show that our proposal is effective in terms of gas cost in Ethereum.

Keywords: Outsourcing Computation, Blockchain, Polynomial Computation.

1 Introduction

Cloud computing provides a cost-effective, flexible, and on-demand avenue for accessing its centralized pool of computational resources. One of its most compelling advantages lies in the outsourcing computing paradigm, enabling resource-constrained clients, utilizing lightweight devices such as smartphones and laptops, to offload extensive computation tasks to the robust cloud infrastructure on a pay-per-use basis. Consequently, both enterprises and individuals can sidestep substantial investments in hardware and software deployment and maintenance. Despite these considerable benefits, the realization of outsourcing computing falls short of expectations due to notable security challenges. A primary obstacle to this technological transformation is *Integrity*, wherein the data owner relinquishes direct control over computation tasks once outsourced to the cloud. In the absence of supervision, the cloud may potentially provide falsified results to conserve computational resources or reduce response time. This also directly ties into the broader landscape of mobile internet security, as data integrity is a fundamental concern in the secure operation of mobile devices connected to the internet.

To address this challenge, Gennaro et al. [1] introduced the concept of Verifiable Outsourcing Computation (VC) to ensure the robust correctness of results. The key to VC is that the

The 7th International Conference on Mobile Internet Security (MobiSec'23), Dec. 19-21, 2023, Okinawa, JAPAN, Article No. 31

*Corresponding author: Software College, Northeastern University, Shenyang, Heping District, Beijing 110819, China, E-mail: wangqiang1@mail.neu.edu.cn.

client's computational cost of verification should be markedly lower than that of local execution. Otherwise, the client would logically opt for in-house computation. After this seminal proposal, numerous VC schemes [2, 3, 4] have been advanced. Based on function types, VC can be broadly categorized into general and specific classes. While theoretically applicable to any function, the former often entails intricate processes such as zero-knowledge proofs and fully homomorphic encryption, rendering it principally of theoretical significance. Conversely, the latter is designed for specific function classes and exhibited significantly enhanced efficiency in practical applications.

This paper predominantly centers on a specific Verifiable Computation (VC) scheme, specifically for polynomial evaluation. Polynomial evaluation, a foundational mathematical operation, finds widespread application in real-world scenarios such as data analysis and machine learning. It is noteworthy as a crucial step towards achieving efficient general VC, wherein all computations can be encoded into arithmetic circuits and subsequently transformed into polynomials. The significance of polynomial evaluation is underscored by the multitude of VC schemes proposed over the past decades. However, the majority of existing schemes primarily emphasize ensuring that the client obtains a valid result before payment, often neglecting the interests of the cloud service provider. In such instances, the client may attempt to acquire the result without payment by intentionally disputing the correctness of the cloud's computation result, even if the computation was performed honestly. Conversely, the cloud might intentionally provide inaccurate results, opting for suboptimal use of resources and time to maximize economic gains. In cases of disputes, resolution involves the introduction of a fully trusted third party (TTP), acting as an adjudicator. This ex-post measure resolves disputes, but the process is not immediate, and real-world scenarios often lack a reliable TTP. To address this challenge, Guan et al. [5], leveraging blockchain technology, proposed the first and only fair outsourcing polynomial computation (FOPC) without a TTP. They employed Horner's method and blockchain to eliminate the need for a TTP. Despite being state-of-the-art, this approach introduces new security challenges, as outlined below.

1. **Privacy:** This scheme directly outsources the polynomial $f(x)$ to the malicious cloud in a *plaintext* manner. The data owner does not wish to expose the privacy of the outsourced function $f(x)$, input and output against the cloud since the they may contain sensitive information. A naive solution to achieve privacy is to additionally envelop it under a fully homomorphic encryption (FHE) scheme, as mentioned in [5]. However, privacy-preserving verifiable outsourcing computation for polynomials so obtained is inefficient in practice since it requires the resource-limited client to perform heavy FHE operations, which is far less efficient than regular encryption. Meanwhile, it also requires the cloud to execute expensive computation over encrypted data. Therefore, there is a pressing need for privacy-preserving VC schemes for polynomials without FHE.

2. **False positive rate:** This scheme employs Horner's method and blockchain to assure the correctness of the computation result returned by the cloud. However, the false positive rate (FPR) [1] increases linearly to the number of r_i's to be sampled, where $r_i = a_{n-i-1} + x \times r_{i-1}$ is the i-th linear function in Horner's method[2]. Therefore, the checkability rate can not reach up to 100%. For example, suppose the intermediate results r_1, r_2, and r_3 are forged while the consecutive results r_4, r_5, and r_6 are computed correctly based on these wrong results. Sampling r_5 and r_6, it will pass the verification algorithm executed by smart contracts. The smart contract will automatically transfer the funds to the cloud.

[1]The probability of verification failure

[2]a_{n-i-1} is the $n-i-1$-th coefficient of the outsourced polynomial $f(x)$

In this case, the client still has to pay despite getting the wrong result. Therefore, there is a pressing need for VC schemes for polynomials without any FPR.

To resolve the problems mentioned above, we propose a novel blockchain-based privacy-preserving fair outsourcing polynomial computation (PFOC) scheme without FHE and FPR. Without FHE, it not only preserves the privacy of the function of the data owner but also protects the confidentiality of inputs/outputs of the client. Unlike FOPC, our proposed scheme PFOC can achieve absolute fairness due to no FPR.

1.1 Our Results and Contributions

Our results and contributions can be summarized as follows.

1. We protect the privacy of the polynomial using the Paillier encryption. Due to its homomorphism, the cloud server can perform computations while learning nothing about the polynomial. To hide the input without using expensive FHE, the true input is blinded with a random number. In the end, the client can recover the true result and the cloud can not get any information about the input and output.

2. To achieve no FPR, we propose a deterministic verifiable mechanism. To achieve public verifiability, we employ SGX to store the verification key in the SGX enclave of miners. As a result, any miner can run the smart contract to check the correctness of the result.

3. We implement a prototype of our proposed scheme and give a comprehensive comparison. The experimental results show that our protocol is more practical.

1.2 Related Work

Verifiable outsourcing computation (VC) is a significant paradigm that allows a computationally weak client to securely delegate some complex computations to a powerful but untrusted cloud. In this setting, the client can check the correctness of the outsourced computation with little cost. This primitive was originally introduced by Gennaro et al. [1]. Following this pioneering work, plenty of VC schemes have been proposed. According to the employed verification mechanism, VC can be classified into two-fold: publicly VC (PVC) and privately VC. The former does not involve the data owner's secret key during the verification phase, allowing anyone including the data owner to check whether the cloud returns a valid result. In contrast, the latter restricts result integrity verification to the data owner who possesses the secret key. Therefore, the latter is a two-party protocol that includes the data owner and the cloud. In general, privately VC is significantly more efficient than PVC. However, privately VC poses challenges for the cloud, as the client may attempt to get the result without any payment by deliberately accusing their misbehavior, even if the cloud conducts the computation honestly. In such disputes, data owners are understandably hesitant to disclose their secret keys to substantiate their claims. To this end, the concept of publicly VC was first proposed by Parno et al. in 2012 [6]. In the following, we will mainly focus on publicly verifiable outsourcing computation.

Polynomial computation is a fundamental mathematical operation, which is widely used in many applications. Due to its importance, numerous PVC schemes for polynomials have been proposed in the past decades. Fiore et al. [7] gave a new outsourcing polynomial computation scheme based on the bilinear map, while the verification of the resulting scheme is not efficient as expected. Later on, Catalano et. al. [8], Backes et al. [9], Fiore et al. [10], and Zhang et al. [11] present new schemes with more efficient verification by using different techniques, such as homomorphic hash function and homomorphic message authentic codes. The existing schemes [12, 13, 11] also allow public verification. [12] proposed the signatures of correct computation model (SCC), the worker would produce a succinct signature to vouch for the correctness of the result, and it could be verified efficiently.[13] proposed a publicly verifiable computation of polynomial based on the concept of the Euclidean division. [11] employed a linearly homomorphic private-key encryption scheme to achieve public verifiability. However, none of them considered the privacy of the input or the function.

Verifiable outsourcing computation ensures the integrity of computation results, but achieving fairness in the interests of both users and cloud servers remains a challenge. When a dispute arises between a user and a cloud server, a complex process involving a trusted third party (TTP) will be required. Trusted third parties (TTP) are a common way of resolving disputes over the correctness of computational results in cloud computing, but they have some shortcomings. Because TTP requires verification and arbitration of transactions, it cannot respond quickly to disputes. In addition, if a TTP fails, the entire system is affected and there is a single point of failure problem. With the development of blockchain, it is possible to apply blockchain to outsourced computing. Kumaresan et al. [14] proposed a model to motivate correct computing in the Bitcoin network. In their system, the worker should pay a deposit that would be lost if the result cannot pass the verification. Their scheme mainly focused on the timely delivery of the results and the fairness of the payment. Campanelli et al. [15] realized the zero-knowledge payment of services based on blockchain, but their scheme lacks design details, and its efficiency is not as good as expected. Based on game theory and smart contracts, Dong et al. [16] proposed an efficient verifiable outsourcing computing solution. However, their scheme only works when the workers do not collude with each other. This assumption is a little bit strong in the blockchain system. For example, more than 15% of mining power may come from the same mining pool in Bitcoin. In other words, there is more than 15% chance that the two workers are from the same organization. Krol et al. [17] proposed an efficient and secure blockchain-based outsourcing computation solution by using trusted hardware that would increase the user's cost. Lin et al. [18] studied how to use blockchain to secure outsourcing bilinear pairings. Zhang et al. [19] employed a challenge-and-proof manner to build a fair payment framework for outsourcing services, but it did not discuss the approach of constructing correctness proof for specific computational tasks. Cui et al. [20] proposed an outsourced decryption scheme for a functional encryption scheme, in which a decrypted result is first verified by the user and further verified by miners if the user rejects the result. Guan et al. [5] proposed a new scheme for outsourced polynomial computation based on blockchain technology [21] and Horner's law, and proved the fairness of the scheme through game theory analysis. However, this state-of-the-art still suffers from some new security challenges, which will slow down or impede its promotion and popularization. Firstly, it exposes privacy vulnerabilities by directly outsourcing the polynomial $f(x)$ to a malicious cloud in plaintext. Besides, all orignial inputs and outputs without any protection will be exposed as well. The second one is the lower checkability rate due to the sampling technique, which is linear to the number of selected immediate results. Therefore, there is a pressing need for a privacy-preserving VC polynomial scheme without overwhelming checkability rate.

1.3 Paper Organization

The rest of this paper is organized as follows. Section 2 reviews some knowledge that is needed beforehand. Section 3 formally defines our proposed system model and its security.

Section 4 provides a concrete construction. Section 5 provides a detailed theoretical analysis and simulation. Finally, Section 6 concludes the paper.

2 Preliminaries

2.1 SGX

Intel software guard extensions (SGX)[22], as trusted hardware technologies developed by Intel Corporation, have been integrated into Intel's commodity CPUs and provide the possibility of large-scale usage. SGX offers a secure container by leveraging trusted hardware. Remote clients can upload the code and data into the secure container, and the process can be proven reliable. The secure container protects the confidentiality and integrity of data while the computation is being performed on it. The code and data loaded in the secure container cannot be tampered with by the outside world. At the same time, the built-in services (e.g., trusted random number, trusted time, and trusted monotonic counter) also provide powerful assurance for designing protocols[23]. Benefiting from the above characteristics, SGX technology can provide powerful support to solve the dilemma in the blockchain. Thus, utilizing a powerful tool such as SGX in the blockchain area has become a new research direction.

2.2 Paillier Cryptosystem

A $\mathsf{Paillier}$ scheme consists of the following algorithms:

$(\mathsf{Paillier.PK}, \mathsf{Paillier.SK}) \leftarrow \mathsf{Paillier.KeyGen}(1^\kappa)$: The key generation algorithm takes as input the security parameter κ and outputs a key pair $(\mathsf{Paillier.PK}, \mathsf{Paillier.SK})$.

$c \leftarrow \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(m)$: The encryption algorithm takes as inputs the message m and the public key $\mathsf{Paillier.PK}$ and outputs the ciphertext c.

$m \leftarrow \mathsf{Paillier.Dec}_{\mathsf{Paillier.SK}}(c)$: The decryption algorithm takes as inputs the ciphertext c and the secret key $\mathsf{Paillier.SK}$ and outputs the message m.

The cryptosystem supports the following operations, which can be performed without knowledge of the private key:

1. Given the encryption of m_0 and m_1, c_0 and c_1, we can efficiently compute the encryption of $a + b$, denoted:
$$\mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(m_0 + m_1) = c_0 \otimes c_1,$$
2. Given a constant c and the encryption of m_0, c_0, we can efficiently compute the encryption of cm_0, denoted:
$$\mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(c \cdot m_0) = c_0^c,$$
where $\otimes$ is the operation over the ciphertext.

2.3 IPFS

Inter-Planetary File System (IPFS) is proposed by Benet [24] and developed by Protocol Labs. It aims to provide users a resilient peer-to-peer file system for big file storage and sharing similar to BitTorrent. Meanwhile, it additionally supports content-addressing and version-controlling properties by using distributed hash tables and git, respectively. With the former property, users can obtain and verify the data easily and quickly. The latter property enables users to review the old version of the data.

Combination of Blockchain and IPFS. As blockchain is originally designed to be a public ledger to record transactions, most blockchain systems adopted many approaches to encourage the transaction size to be small to ensure the network performance. Consequently, it is either impossible or expensive to store big files directly in blockchain systems. Hence, as a distributed file system, IPFS has become a popular solution for data and resource storage in blockchain-based distributed applications (DApps) [25, 26, 27]. Specifically, the DApp can store data into IPFS and keep the corresponding addresses in the blockchain. Then, users can retrieve the corresponding data or files from IPFS with the addresses obtained from the DApp. On the other hand, when the DApp needs to read a block of data from IPFS, it can request through Oracle services. In this paper, we consider a decentralized oracle service providing IPFS data for smart contracts.

2.4 Polynomial Congruence Theorem

Theorem 1. *Given $f(x)$, where $x \in Z$, If the value of the polynomial $f(u) \geq 0, u \in Z$, then the following equation always holds:*

$$f(u+r) \mod r = f(u)$$

where the integer $r \geq f(u)$.

Due to the space limitation, we refer the interested reader to [4] for rigorous proof. This theorem will be used to hide the true input x in our construction, since the dummy input $x + r$ and true input x are indistinguishable. In the end, the data owner can recover the final result $f(x)$ by making $f(x + r)$ modular r.

3 Privacy-preserving Fair Outsourcing Computation Based on Blockchain

In this section, we formalize the definition of the privacy-preserving fair outsourced computation model based on blockchain and its security required in our scheme.

3.1 Model Definition

A privacy-preserving fair outsourced computation model based on blockchain comprises four different entities which are illustrated in Fig. 1. We describe them as follows:

1. Data owner: an untrusted entity with limited computation capability, who is responsible for delegating the computation task to the cloud. When he receives the final result related to his input, he may refuse to pay by claiming that the answer is incorrect or not received.

2. Cloud: an untrusted entity with powerful computation resources, who is responsible for conducting the computation task in a pay-per-use manner. He has the potential to get a payment when he returns a wrong result.

3. Blockchain: a fully trusted entity, who is composed of many miners with SGX. The miners execute the smart contract to verify the correctness of the result returned by the cloud. If the verification passes, it forwards the reward to the cloud server. Otherwise, it returns the deposit to the client.

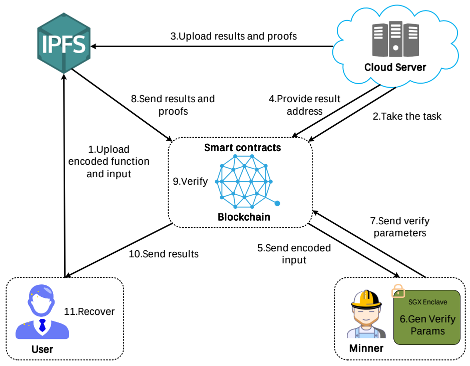


Figure 1: The Architecture of PFOC

4. IPFS: a fully trusted entity, who is responsible for recording the final result and witness provided by the cloud.

In the end, the malicious cloud can get any information about the outsourced function f, the input x, and output $f(x)$. When a dispute occurs, it can be fairly addressed immediately.

Definition 1 (Privacy-preserving fair outsourcing computation based on blockchain)**.** *The privacy-preserving fair outsourcing computation based on blockchain is comprised of the following seven algorithms.*

Setup $\left(1^{\lambda}, F\right) \rightarrow (PK, SK)$*: The setup algorithm is run by the data owner. It takes as input the security parameters λ and a family of functions F and outputs a key pair (PK, SK).*

EncFunction$(PK, SK, f) \rightarrow \sigma_f$*: The function encoding algorithm is run by the data owner. It takes as input the public key PK, the private key SK and the function $f \in F$, and outputs the encoded function σ_f.*

EncInput$(PK, x) \rightarrow \sigma_x$*: The input encoding algorithm is run by the data owner. It takes as input the public key PK and the input x, and outputs the encoded input σ_x.*

Compute$(PK, \sigma_f, \sigma_x) \rightarrow (\sigma_y, \pi)$*: The computation algorithm is run by the untrusted cloud. It takes as input the public key PK, the encoded function σ_f and encoded input σ_x, and outputs the encoded output σ_y and witness π.*

ParamGen $(PK, SK, \sigma_x, \sigma_y) \rightarrow param$*: The verification parameter generation algorithm is run by the miner's* SGX*. It takes as input the public key PK, private key SK, encoded input σ_x and encoded output σ_y, and outputs verification parameter $param$.*

Verify $(PK, \pi, \sigma_y, param) \rightarrow \{0 \text{ or } 1\}$*: The verification algorithm is run by the smart contract on the blockchain. It takes as input the public key PK, witness σ_y, encoded output σ_y and verification parameter $param$. It outputs 1 if verification is successful and 0 otherwise.*

Recover $(SK, PK, x, \sigma_y) \rightarrow y$*: The recovery algorithm is run by the data owner. It takes as input the public key PK, the private key SK, the true input x and encoded output σ_y, and outputs the true result $y = f(x)$.*

3.2 Correctness and Security Definitions

Intuitively, a privacy-preserving fair outsourced computation based on blockchain scheme is correct if whenever its algorithms are executed honestly, a valid result will never be rejected and could be recovered. More formally:

Definition 2 (Correctness)**.** *A privacy-preserving fair outsourcing computation based on blockchain scheme is correct if the following holds:*

$$\Pr\left[\begin{array}{l}\mathsf{Setup}\left(1^{\lambda}, F\right) \rightarrow (PK, SK); \\ \mathsf{EncFunction}(PK, SK, f) \rightarrow \sigma_f; \\ \mathsf{EncInput}(PK, x) \rightarrow \sigma_x; \\ \mathsf{Compute}(PK, \sigma_f, \sigma_x) \rightarrow (\sigma_y, \pi); \\ \mathsf{ParamGen}\left(PK, SK, \sigma_x, \sigma_y\right) \rightarrow param: \\ \mathsf{Verify}\left(PK, \pi, \sigma_y, param\right) \rightarrow 1 \wedge \\ \mathsf{Recover}\left(SK, PK, x, \sigma_y\right) \rightarrow y\end{array}\right] \geq 1 - negl(\lambda)$$

Intuitively, a privacy-preserving fair outsourcing computation scheme based on blockchain satisfies result unforgeability if whenever arbitrary adversary $\mathcal{A}$ cannot convince a verifier to accept a wrong result with overwhelming probability. More formally:

Definition 3 (Result Unforgeability)**.** *Let* $\prod$ *be a privacy-preserving fair outsourced computation based on blockchain scheme, and let* $\mathcal{A}$ *be a PPT machine. We define result unforgeability via the following experiment* $\mathrm{Exp}_{\mathcal{A}}^{\mathsf{RU}}\left[\prod, F, \lambda\right]$.

```
$\mathrm{Exp}_{\mathcal{A}}^{\mathsf{RU}}[\prod, F, \lambda]$:
$\mathsf{Setup}\left(1^{\lambda}, F\right) \rightarrow (PK, SK)$;
$\mathsf{EncFunction}(PK, SK, f) \rightarrow \sigma_f$;
From $i = 1$ to $d$:
    $\mathcal{A}\left(PK, x_1, \sigma_{x_1}, s, x_{i-1}, \sigma_{x_{i-1}}\right) \rightarrow x_i$;
    $\mathsf{EncInput}\left(PK, x_i\right) \rightarrow \sigma_{x_i}$;
$\mathcal{A}\left(PK, \sigma_{x_1} s, \sigma_{x_d}\right) \rightarrow x^*$;
$\mathsf{EncInput}\left(PK, x^*\right) \rightarrow \sigma_{x^*}$;
$\mathcal{A}\left(PK, \sigma_f, \sigma_{x_1}, s, \sigma_{x_d}, \sigma_{x^*}\right) \rightarrow (\sigma'_y, \pi)$;
$\mathsf{ParamGen}\left(PK, SK, \sigma_{x^*}, \sigma'_y\right) \rightarrow param$;
$\mathsf{Verify}\left(PK, \pi, \sigma'_y, param\right) \rightarrow b$;
$\mathsf{Recover}\left(SK, PK, x, \sigma'_y\right) \rightarrow y'$;
If $b = 1$ and $y' \neq f\left(x^*\right)$:
  output 1;
else
  output 0;
```

For any $\lambda \in \mathbb{N}$*, we define the advantage of arbitrary* $\mathcal{A}$ *in the above experiment as*

$$Adv_{\mathcal{A}}^{\mathsf{RU}}\left(\prod, F, \lambda\right) = \Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathsf{RU}}[\prod, F, \lambda] = 1\right]$$

We say that PFOC *achieves result unforgeability if* $Adv_{\mathcal{A}}^{\mathsf{RU}}\left(\prod, , \lambda\right) \leqslant negl\left(\lambda\right)$.

Intuitively, a privacy-preserving fair outsourcing computation based on blockchain scheme satisfies input privacy if whenever arbitrary adversary $\mathcal{A}$ cannot distinguish a true input from a dummy input.

Definition 4 (Input Privacy)**.** *Let* $\prod$ *be a privacy-preserving fair outsourced computation based on blockchain scheme, and let* $\mathcal{A}$ *be a PPT machine. We define input privacy via the following experiment* $\mathrm{Exp}_{\mathcal{A}}^{\mathsf{Privx}}\left[\prod, F, \lambda\right]$.

```
$\mathrm{Exp}_{\mathcal{A}}^{\mathsf{Privx}}[\prod, , \lambda]$:
$\mathsf{Setup}\left(1^{\lambda}, F\right) \to (PK, SK)$;
$\mathcal{A}(PK) \to x_0, x_1$;
$\{0,1\} \xrightarrow{R} b$;
$\mathsf{EncInput}(PK, x_b) \to \sigma_{x_b}$;
$\mathcal{A}(PK, x_0, x_1, \sigma_{x_b}) \to \hat{b}$;
If $\hat{b} = b$:
    output 1;
else
    output 0;
```

For any $\lambda \in \mathbb{N}$, we define the advantage of arbitrary $\mathcal{A}$ in the above experiment as

$$Adv_{\mathcal{A}}^{\mathsf{Privx}}(\prod, F, \lambda) = \left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathsf{Privx}}[\prod, F, \lambda] = 1\right] - \frac{1}{2}\right|$$

We say that PFOC *achieves input privacy if $Adv_{\mathcal{A}}^{\mathsf{Privx}}(\prod, F, \lambda) \leqslant negl(\lambda)$.*

Due to the limitation of space, we omit the definition of output privacy since it is similar to that of input privacy definition.

Intuitively, a privacy-preserving fair outsourcing computation scheme based on blockchain satisfies function privacy if whenever arbitrary adversary $\mathcal{A}$ cannot distinguish between a true function and a dummy function.

Definition 5 (Function Privacy)**.** *Let $\prod$ be a privacy-preserving fair outsourced computation scheme based on blockchain, and let $\mathcal{A}$ be a PPT machine. We define function privacy via the following experiment* $\mathrm{Exp}_{\mathcal{A}}^{\mathsf{Privf}}[\prod, F, \lambda]$.

```
$\mathrm{Exp}_{\mathcal{A}}^{\mathsf{Privf}}[\prod, F, \lambda]$:
$\mathcal{A}(F, \lambda) \to f_0, f_1$, where $|f_0| = |f_1|$;
$\{0,1\} \xrightarrow{R} b$;
Setup $\left(1^{\lambda}, f_b\right) \to (PK_b, SK_b)$;
$\mathsf{EncFunction}(PK_b, SK_b, f_b) \to \sigma_{f_b}$;
$\mathcal{A}(PK_b, f_0, f_1, \sigma_{f_b}) \to \hat{b}$;
If $\hat{b} = b$:
    output 1;
else
    output 0;
```

For any $\lambda \in \mathbb{N}$, we define the advantage of arbitrary $\mathcal{A}$ in the above experiment as

$$Adv_{\mathcal{A}}^{\mathsf{Privf}}(\prod, F, \lambda) = \left|\Pr\left[\mathrm{Exp}_{\mathcal{A}}^{\mathsf{Privf}}[\prod, F, \lambda] = 1\right] - \frac{1}{2}\right|$$

We say that PFOC *achieves function privacy if $Adv_{\mathcal{A}}^{\mathsf{Privf}}(\prod, f, \lambda) \leqslant negl(\lambda)$.*

4 The Construction Of Privacy-preserving Fair Outsourcing Polynomial Computation Based on Blockchain

4.1 Details of Our Construction

$\mathsf{Setup}\left(1^{\lambda}, F\right) \rightarrow (PK, SK)$: Given a security parameter λ and a family of polynomial F, the data owner first uniformly choose $d \in [n]$, $c, r_0, r_1, r_2 \in Z_p^*$. Next, it generates a key pair $(\mathsf{Paillier.PK}, \mathsf{Paillier.SK})$ by conducting $\mathsf{Paillier.KeyGen}$ algorithm. Finally, it publishes the public key $PK = \{\mathsf{Paillier.PK}, max = \max(F)\}$ and keeps the private key $SK = \{\mathsf{Paillier.SK}, d, c, r_0, r_1, r_2\}$ secret.

$\mathsf{EncFunction}(PK, SK, f) \rightarrow \sigma_f$: The data owner first parses the public key PK and private key SK as $\{\mathsf{Paillier.PK}, max\}$ and $\{\mathsf{Paillier.SK}, c, d, r_0, r_1, r_2\}$, respectively. Next, it generates two polynomials $r(x)$ and $g(x)$ as follows:

$$r(x) = r_0 x^d + r_1 + r_2 x \tag{1}$$

$$g(x) = c \cdot f(x) + r(x) = \sum_{i=0}^{n} b_i x^i \tag{2}$$

Then, the data owner executes $\mathsf{Paillier.Enc}$ algorithm to encrypt outsourcing polynomials $f(x)$ and gets the encrypted polynomial $f'(x)$ as follows.

$$f'(x) = \sum_{i=0}^{n} a_i' x^i, \tag{3}$$

where $a_i' = \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(a_i)$. Similarly, it can get the encrypted polynomial $g'(x)$ as follows.

$$g'(x) = \sum_{i=0}^{n} b_i' x^i, \tag{4}$$

where $b_i' = \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(b_i)$. Finally, it sets the encoded polynomial $\sigma_f = \{f'(x), g'(x)\}$ and uploads σ_f to IPFS.

$\mathsf{EncInput}(PK, x) \rightarrow \sigma_x$: The data owner first parses the public key PK as $\{\mathsf{Paillier.PK}, max\}$. Next, it uniformly chooses a random $r > max$ and computes the encoded input

$$\sigma_x = x + r. \tag{5}$$

Finally, it uploads σ_x to the IPFS.

$\mathsf{Compute}(PK, \sigma_f, \sigma_x) \rightarrow \{\sigma_y, \pi\}$: Upon receiving the encoded input σ_x, the cloud generates σ_y and π by computing

$$\sigma_y = f'(\sigma_x) = \prod_{i=0}^{n} \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(a_i)^{\sigma_x^i} \tag{6}$$

$$\pi = g'(\sigma_x) = \prod_{i=0}^{n} \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(b_i)^{\sigma_x^i} \tag{7}$$

Finally, the cloud uploads the encoded result σ_y and witness π to IPFS.

$\mathsf{ParamGen}\left(PK, SK, \sigma_x, \sigma_y\right) \rightarrow param$: With the secure channel established by the successful remote authentication of SGX, the client can send the private key SK to the miner's SGX

securely. After receiving it, the enclave parses SK as $\{\mathsf{Paillier.SK}, c, d, r_0, r_1, r_2\}$. Next, it generates the polynomial $r(x) = r_0x^d + r_1 + r_2x$ and computes $R = r(\sigma_x)$. Then, it generates R' by computing

$$R' = \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(R) = \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(r(\sigma_x)) \tag{8}$$

Afterward, it computes

$$Y' = \sigma_y^c. \tag{9}$$

Finally, it sets verification parameter $param = \{Y', R'\}$ and submits $param$ to the smart contract.

$\mathsf{Verify}\,(PK, \pi, \sigma_y, param) \rightarrow \{0 \text{ or } 1\}$: Upon receiving the encoded result σ_y and witness $param$, the smart contract parses the verification parameter $param$ as $\{Y', R'\}$. Next, it checks whether the following equation holds:

$$\pi = Y' \cdot R' \tag{10}$$

If not, it outputs 0 and aborts; Otherwise, it outputs 1 and accepts the encoded result σ_y.

$\mathsf{Recover}\,(SK, PK, x, \sigma_y) \rightarrow y$: The data owner first parses the private key SK and public key PK as $\{\mathsf{Paillier.PK}, max\}$ and $\{\mathsf{Paillier.SK}, c, d, r_0, r_1, r_2\}$, respectively. Next, it recovers encoded result y_1 by computing

$$y_1 = \mathsf{Paillier.Dec}_{\mathsf{Paillier.SK}}(\sigma_y) = f(x + r) \tag{11}$$

Finally, it recovers the final result y by computing

$$y = y_1 = f(x + r) \mod r \tag{12}$$

Remark 1. *Full homomorphic encryption can also be used for the polynomial coefficients f and the input x hidden, we call this scheme FHPC. The basic process is the same as our proposed scheme. When the data owner performs the function encoding and input encoding algorithm, the two polynomial coefficients, input and verification parameters are fully homomorphic encrypted. When the client performs the result recovery algorithm, the result $f(x)$ is obtained by decrypting it using the fully homomorphic private key.*

4.2 Correctness

According to Definition 2, to prove the correctness, we only need to argue that the valid encoded result σ_y can pass Verify algorithm and can be decoded to the final result $y = f(x)$ if all the entities involved are honest.

For the first part, we will argue it mainly based on Equation 10. According to Equations 9 and 8, the right-hand side of Equation 10 can be expressed as below:

$$\begin{aligned} Y' \cdot R' &= \sigma_y^c \cdot \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(R\right) \\ &= \sigma_y^c \cdot \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(r(\sigma_x)\right) \end{aligned}$$

According to Equation 6, the above equation can be re-written as follows:

$$\begin{aligned}
Y' \cdot R' &= \prod_{i=0}^{n} \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(c \cdot a_i)^{\sigma_x^i} \cdot \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(r\left(\sigma_x\right)\right) \\
&= \prod_{i=0}^{n} \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}(c \cdot a_i \cdot \sigma_x^i) \cdot \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(r\left(\sigma_x\right)\right) \\
&= \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(\sum_{i=0}^{n} c \cdot a_i \cdot \sigma_x^i\right) \cdot \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(r\left(\sigma_x\right)\right) \\
&= \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(c \cdot f\left(\sigma_x\right) + r\left(\sigma_x\right)\right) \\
&= \mathsf{Paillier.Enc}_{\mathsf{Paillier.PK}}\left(g(\sigma_x)\right) \\
&= \pi
\end{aligned}$$

Obviously, if all participants follow honestly all algorithms described above, the valid result σ_y will never be rejected by verification algorithm Verify.

For the second part, we will argue that the encoded result σ_y can be recovered to the final result $y = f(x)$. According to Equation 11, the data owner can get $y_1 = f(\sigma_x) = f(x + r)$ by conducting $\mathsf{Paillier.Dec}$. According to Theorem 1, we can get $f(x) = f(x{+}r) \mod r$. Therefore, the Equation 12 holds.

From all the above, our proposed scheme achieves correctness.

5 Evaluation

In this section, we first theoretically compare our proposed scheme with Guan et al.'s scheme [5] (we call it as FOPC), Ye et al.'s scheme [28] (we call it as VDPC), and privacy-preserving PFOC using fully homomorphic encryption scheme (we call it as FHPC) in two folds: the desired properties and the computation complexity. Then, we provide a prototypal implement to compare their computation cost.

5.1 Properties

Table 1 summarizes the comparison of desired properties. Only VDPC requires a trusted third party (TTP) to handle disputes between the data owner and the cloud, while other schemes utilize smart contracts (deployed in the decentralized blockchain) to automatically check the correctness of the computation result. The TTP can ensure that the dispute will be finally addressed, but it can not make responses immediately like other schemes. Meanwhile, TTP-based schemes also suffer from one single-point failure. FOPC focuses on how to efficiently obtain the fairness of the outsourced polynomial computation without considering privacy, and VDPC protects the privacy of the output. FHPC achieves privacy using FHE, so our proposed scheme is more efficient than FHPC. Only VDPC involves the private key in the verification phase, so VDPC is a privately verifiable outsourcing computation. As mentioned in section 1, public verifiability is the precondition of fairness. In the simulation, we will omit VDPC due to its private verifiability. Using the sampling technique, FPR both in FOPC and FHPC schemes decreases linearly to the number of the chosen points. Hence, their FPR is higher than our proposed scheme. FHPC has the same desired properties as ours, but their efficiency is lower than ours. Overall, our proposed scheme is more practical.

Table 1: Comparison of Desired Properties

Scheme	Decentration	Privacy			Public Verifiablity	FPR	Fairness
		Polynomial	Input	Output			
FOPC[5]	✓	×	×	×	✓	High	High
VDPC[28]	×	×	×	✓	×	Low	Low
FHPC	✓	✓	✓	✓	✓	High	High
Our proposal	✓	✓	✓	✓	✓	Low	High

Table 2: Comparison of Computational Cost

Scheme	Setup	EncFunction	EncInput	Compute	ParamGen	Verify	Recover
FOPC[5]	N/A	N/A	N/A	n Mul+1Mod	N/A	sMul	N/A
VDPC[28]	1 Mul	(n+2) Mul+1 Exp	N/A	2n Mul+2Mod	1 Exp+1 Mul+1 Mod	1Mul+1Mod	1Mod
FHPC	FH.Setup	(n+1)FH.Enc	1FH.Enc	nFH.Mul+nFH.Add+1Mod	N/A	sFH.Mul	1FH.Dec
Our proposal	P.Setup	(n+2) Mul+1 Exp+2(n+3)P.Enc	1 Add	2n Mul+(4n-2) Exp+2 Mod	2Exp+1Mul+1P.Enc+1Mod	1Mul+1Mod	1P.Dec+2Mod

5.2 Computation Cost

Since the computation cost is mainly determined by the exponentiation, multiplication, addition, modulo operation, Paillier and full homomorphism encryption algorithms, we evaluate it by counting the number of these operations. Table 2 summarizes the comparison of the computation costs of the schemes mentioned above. In Table 2, we denote the time cost of exponentiation by Exp, the time cost of multiplication by Mul, the time cost of addition by Add, the time cost of modulo operation by Mod, the time cost of key generation, encryption and decryption algorithms of Pallier Cryptosystem by P.Setup, P.Enc, P.Dec respectively. Similarly, the time cost of these three algorithms of fully homomorphic encryption by FH.Setup, FH.Enc, and FH.Dec, respectively. We denote the number of selected intermediate results by s. In EncInput phase, our proposed scheme is much more efficient than FHPC since it only requires one single addition operation rather than an encryption operation of FHE. In terms of time cost of Compute, our proposed scheme is similar to FHPC. FOPC is the most efficient since the cloud does not conduct additional computation to guarantee the integrity of the result. VDPC has to conduct another polynomial with the same size as the outsourcing polynomial. For Verify algorithm, the time cost of FOPC is linear to the number s of selected intermediate results, while other schemes are all independent of s. In the Recover phase, our proposed scheme is much more efficient than FHPC since the decryption algorithm of FHE is much more expensive than that of Paillier.

5.3 Simulation

In this section, we provide a thorough experimental evaluation of our proposed scheme. To precisely evaluate the computation cost at client, cloud server, miner and Ethereum, all simulations were conducted on Ubuntu 23.03 virtual machine simulating an Intel(R) Core(TM) i7-12700HQ CPU @ 2.10GHz processor and 8GB memory. Then, to evaluate the gas consumption during the Verify phase, we tested the Solidity implementation on REMIX. The performance of the proposed scheme was demonstrated for modulo lengths $p = 64, 128, 256, 512, 1024$ bits, polynomial terms $n = 100, 200, 300, 400, 500, 600$ and a Paillier public key length of 1024 bits.

Figure 2a depicts the comparison of time cost in the Setup phase. VDPC is the most efficient scheme, while FHPC costs the most expensive overhead. The computational cost of our scheme PFOC is larger because of the extra time spent on Paillier.KeyGen. Figure 2b depicts the comparison of time cost in the EncFunction phase. The time cost of all these schemes is linearly related to the number of polynomial terms n. The computational cost of PFOC is

Table 3: Comparison of Gas Cost

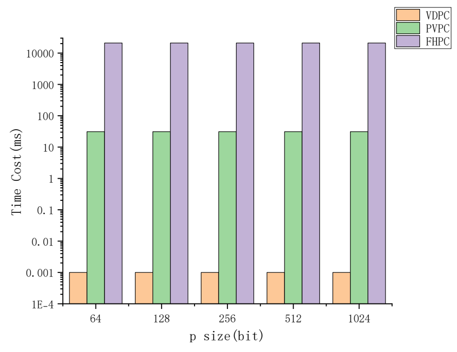
Scheme	Degree	gas consumption	Amount($)
Polynomial Compute	100	16,509,639	907.82
	200	41,557,187	2 269.56
	300	67,658,987	3 685.77
	400	93,760,787	5 101.97
	500	119,862,587	6 536.33
	600	145,964,387	7 952.54
FOPC Verify	10	2610180	141.62
PFOC Verify		77888	4.25

larger. This is because the two polynomial coefficients have to be encrypted using Paillier.Enc after encoding the polynomial to protect the client's privacy. Figure 2c depicts the comparison of time cost in the Compute phase. FOPC is the most efficient scheme, but it does not protect the client privacy. In contrast, our scheme hides the client's input, polynomial coefficients, and output, while ensuring computational efficiency, although the cost is larger. Figure 2d depicts the comparison of time cost in the GenWitness phase. VDPC is the most efficient scheme, but it does not support public verification. Our scheme is not only minimally affected by p and n, but also have a computational cost of less than 5ms. Figure 2e depicts the comparison of the gas consumption of the Verify phase. Since FOPC and FHPC do not give a specific selection of several intermediate results, so we choose 10 intermediate results for verification. The results show that FHPC has the highest gas consumption. Gas consumption of our scheme is almost independent of the size of p and n. Figure 2f depicts the comparison of time cost in the Recover phase. The cost of our scheme is larger because the computation results have to be decrypted using Paillier.Dec before the modulo operation to get the final result, but the privacy protection and public verifiably of client are achieved. Figure 3 depicts the comparison of the gas consumption between the polynomial computation directly by Ethereum and the Verify phase of PFOC, as shown in the figure, the gas consumption of polynomial computation directly by Ethereum is much larger than that of our Verify phase.

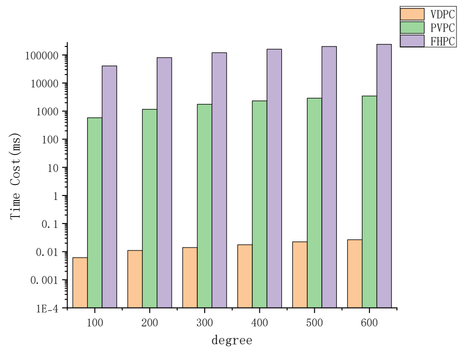
Table 3 shows the gas consumption and cost table of Ethereum's direct polynomial computation under the modulus length $p = 1024$ bits, the verification phase of FOPC and our verification phase. It can be seen from the table and the above experiments that our gas consumption and cost are the least, and Does not vary with the number of polynomial terms and modulus length.

6 Conclusion

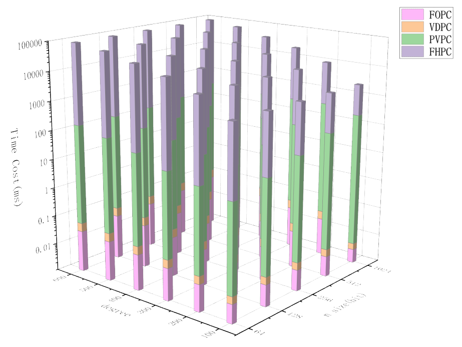
We propose a privacy-preserving fair outsourcing polynomial computation without FPR. To avoid expensive FHE, we utilize the Paillier encryption and blind technique to ensure privacy. We achieve public verifiably using SGX. Besides, our proposed scheme can guarantee fairness with an overwhelming probability. The detailed performance analyses and simulations show that our proposed schemes are more practical in the real world.



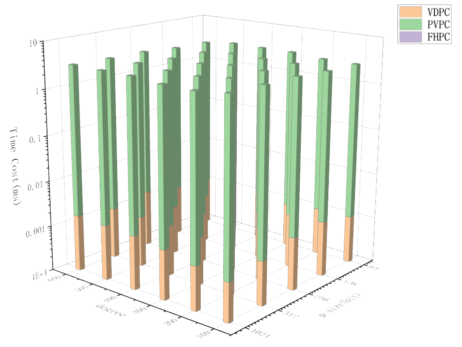
(a) Setup Phase



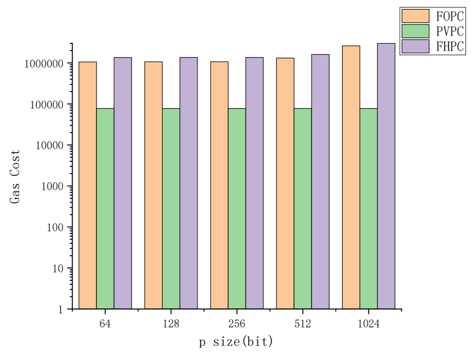
(b) EncFunction Phase



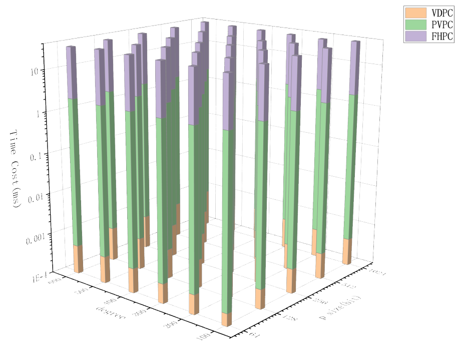
(c) Compute Phase



(d) ParamGen Phase



(e) Verify Phase



(f) Recover Phase

Figure 2: The Comparison of Computation Cost

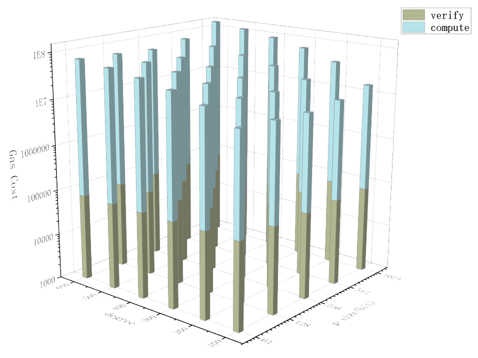


Figure 3: The Comparison of Computation Cost in Gas

Acknowledgement

We thank the anonymous reviewers for their fruitful suggestions. This work was supported in part by the National Natural Science Foundation of China under Grant 62202090 and 62173101, by Liaoning Province Natural Science Foundation Medical-Engineering Cross Joint Fund under Grant 2022-YGJC-24, by Doctoral Scientific Research Foundation of Liaoning Province under Grant 2022-BS-077, by the Fundamental Research Funds for the Central Universities under Grant N2217009, and by State Grid Corporation of China Liaoning Provincial Science and Technology Project under Grant 2023YF-08.

References

[1] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*, pages 465–482. Springer, 2010.

[2] Hyo-Sung Ahn and Young-Hun Lim. Distributed coordination for optimal energy flow in smart grid networks: High-order polynomial approach. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pages 660–665. IEEE, 2016.

[3] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *Annual Cryptology Conference*, pages 111–131. Springer, 2011.

[4] Qiang Wang, Fucai Zhou, Boyang Zhou, Jian Xu, Chunyu Chen, and Qi Wang. Privacy-preserving publicly verifiable databases. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1639–1654, 2022.

[5] Yunguo Guan, Hui Zheng, Jun Shao, Rongxing Lu, and Guiyi Wei. Fair outsourcing polynomial computation based on the blockchain. *IEEE Transactions on Services Computing*, 15(5):2795–2808, 2021.

[6] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings 9*, pages 422–439. Springer, 2012.

[7] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 501–512, 2012.

[8] Dario Catalano, Dario Fiore, Rosario Gennaro, and Konstantinos Vamvourellis. Algebraic (trapdoor) one-way functions and their applications. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 680–699. Springer, 2013.

[9] Michael Backes, Dario Fiore, and Raphael M Reischuk. Verifiable delegation of computation on outsourced data. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 863–874, 2013.

[10] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 844–855, 2014.

[11] Liang Feng Zhang and Reihaneh Safavi-Naini. Protecting data privacy in publicly verifiable delegation of matrix and polynomial functions. *Designs, Codes and Cryptography*, 88(4):677–709, 2020.

[12] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *Theory of Cryptography Conference*, pages 222–242. Springer, 2013.

[13] Kaoutar Elkhiyaoui, Melek Önen, Monir Azraoui, and Refik Molva. Efficient techniques for publicly verifiable delegation of computation. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 119–128, 2016.

[14] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 30–41, 2014.

[15] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 229–243, 2017.

[16] Changyu Dong, Yilei Wang, Amjad Aldweesh, Patrick McCorry, and Aad Van Moorsel. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 211–227, 2017.

[17] Michał Król and Ioannis Psaras. Spoc: Secure payments for outsourced computations. *arXiv preprint arXiv:1807.06462*, 2018.

[18] Chao Lin, Debiao He, Xinyi Huang, Xiang Xie, and Kim-Kwang Raymond Choo. Blockchain-based system for secure outsourcing of bilinear pairings. *Information Sciences*, 527:590–601, 2020.

[19] Yinghui Zhang, Robert H Deng, Ximeng Liu, and Dong Zheng. Outsourcing service fair payment based on blockchain and its applications in cloud computing. *IEEE Transactions on Services Computing*, 14(4):1152–1166, 2018.

[20] Hui Cui, Zhiguo Wan, Xinlei Wei, Surya Nepal, and Xun Yi. Pay as you decrypt: Decryption outsourcing for functional encryption using blockchain. *IEEE Transactions on Information Forensics and Security*, 15:3227–3238, 2020.

[21] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[22] Intel Intel. Software guard extensions programming reference, revision 2, 2014.

[23] Victor Costan and Srinivas Devadas. Intel sgx explained. *Cryptology ePrint Archive*, 2016.

[24] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[25] Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. A trustworthy privacy preserving framework for machine learning in industrial iot systems. *IEEE Transactions on Industrial Informatics*, 16(9):6092–6102, 2020.

[26] Ming Li, Jian Weng, Anjia Yang, Wei Lu, Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, and

Robert H Deng. Crowdbc: A blockchain-based decentralized framework for crowdsourcing. *IEEE transactions on parallel and distributed systems*, 30(6):1251–1266, 2018.

[27] Suayb S Arslan and Turguy Goker. Compress-store on blockchain: a decentralized data processing and immutable storage for multimedia streaming. *Cluster Computing*, 25(3):1957–1968, 2022.

[28] Jun Ye, Haiyan Zhang, and Changyou Fu. Verifiable delegation of polynomials. *Int. J. Netw. Secur.*, 18(2):283–290, 2016.