information

Article

Enhancing Privacy Preservation in Verifiable Computation through Random Permutation Masking to Prevent Leakage

Yang Yang and Guanghua Song *

School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China
* Correspondence: z0004358@zuel.edu.cn

Abstract: Outsourcing computation has become increasingly popular due to its cost-effectiveness, enabling users with limited resources to conduct large-scale computations on potentially untrusted cloud platforms. In order to safeguard privacy, verifiable computing (VC) has emerged as a secure approach, ensuring that the cloud cannot discern users' input and output. Random permutation masking (RPM) is a widely adopted technique in VC protocols to provide robust privacy protection. This work presents a precise definition of the privacy-preserving property of RPM by employing indistinguishability experiments. Moreover, an innovative attack exploiting the greatest common divisor and the least common multiple of each row and column in the encrypted matrices is introduced against RPM. Unlike previous density-based attacks, this novel approach offers a significant advantage by allowing the reconstruction of matrix values from the ciphertext based on RPM. A comprehensive demonstration was provided to illustrate the failure of protocols based on RPM in maintaining the privacy-preserving property under this proposed attack. Furthermore, an extensive series of experiments is conducted to thoroughly validate the effectiveness and advantages of the attack against RPM. The findings of this research highlight vulnerabilities in RPM-based VC protocols and underline the pressing need for further enhancements and alternative privacy-preserving mechanisms in outsourcing computation.

Keywords: outsourcing computation; privacy leakage; verifiable computing; privacy protection; random permutation masking

Citation: Yang, Y.; Song, G. Enhancing Privacy Preservation in Verifiable Computation through Random Permutation Masking to Prevent Leakage. *Information* **2023**, *14*, 603. https://doi.org/10.3390/info14110603

Academic Editor: Leandros Maglaras

Received: 22 September 2023
Revised: 30 October 2023
Accepted: 31 October 2023
Published: 6 November 2023

1. Introduction

The prevalence of outsourcing computation has significantly increased due to the rapid advancements in cloud computing. This practice allows users with limited resources to delegate computationally demanding tasks to robust cloud infrastructures. Consequently, cloud users can achieve substantial cost savings by offloading expensive calculations to potent cloud environments, thereby mitigating the need for substantial investments in hardware and software equipment required for conducting large-scale computations.

The emergence of cloud computing has paved the way for various advantages associated with outsourcing computation. First, cloud service providers possess powerful computing resources and scalable infrastructure that can efficiently handle intricate calculations. This obviates the need for users to expend substantial finances on hardware upgrades or acquire specialized software licenses, resulting in significant cost reduction. Moreover, cloud platforms offer on-demand provisioning, enabling users to access computational resources according to their specific requirements, further minimizing upfront expenses.

Matrix-related computation is widely applied in practical applications. For example, the extraction of facial features in image processing can be implemented using singular value decomposition of matrices [1], and the prediction of data classification in machine learning can be converted into computing linear matrix equations [2]. Unfortunately, these matrix-related calculations might be so large-scale that resource-limited users cannot

complete them in a reasonable time. Thanks to cloud computing, it is an economical alternative for data owners to outsource their matrix-related computations to a powerful cloud. Even if matrix-related calculations are on a moderate scale, outsourcing computation is also preferred for data owners to save computing resources. As a consequence, it is meaningful to design a verifiable computation (VC) protocol for securely outsourcing matrix-related computation to a powerful cloud.

Despite the tremendous benefits, some intrinsic security concerns emerge at the same time [3]. The first crucial problem is whether the results of outsourcing computation are correct. On one hand, software bugs and hardware failures in the cloud might cause miscalculations. On the other hand, the cloud might deliberately inject errors in the computation or simply feedback on a plausible result for computing savings. This concern by cloud users is due to the fact that the computation of outsourced data is out of their control, which is referred to as the security of the outsourcing computation [4]. To meet this challenge, cloud users should have the ability to detect incorrect results. The next significant problem is whether user privacy is exposed to the cloud. Since user data might be sensitive and valuable, e.g., individual photos and business secrets, the cloud might make a profit from selling this information. To address this problem, cloud users need to hide their actual data from the cloud, which is called the privacy of the outsourcing computation [4]. Based on the above discussion, outsourcing computation should satisfy security and privacy requirements. To achieve these two requirements, cloud users have to take some time to perform several local computations. One thing to note is that this time is suggested to be substantially cheaper than cloud users' time to complete the original computations locally. Otherwise, there is no need for cloud users to outsource their computations.

Up to now, there are a large number of VC protocols for securely outsourcing matrix-related computation, including matrix inversion, and matrix–matrix multiplication, to just list a few. Most of them are based on random permutation masking (RPM) [5–12]. The reason is as follows: On one hand, user matrices can be easily encrypted by multiplying some well-designed 1-sparse matrices in which each row and column have only one nonzero element. On the other hand, the computational results of the encrypted matrices can be quickly decrypted by multiplying the inversions of the masking matrices, which is proven to be still 1-sparse. In such a way, cloud users can efficiently utilize RPM to protect the privacy of the outsourced data without performing expensive cryptographic operations. Consequently, RPM is very popular in the design of VC protocols for securely outsourcing matrix-related computation.

There are two natural questions about the existing RPM-based protocols: one is whether the privacy-preserving definition is properly formalized; the other is whether they achieve the privacy-preserving property. Focusing on these two issues, Zhao et al. first utilized equal ratio [13] to design two attacks, which can be utilized to break the privacy-preserving property of the following two types of VC protocols: one is to encrypt the plaintext by performing the diagonal matrix–matrix multiplication; the other is to encode the plaintext by using the random matrix–matrix addition. However, these two attacks cannot be used to crack the privacy-preserving property of the RPM-based VC protocols, which is due to the complex construction of the RPM-based encryption. To meet this challenge, Zhao et al. [14] proposed an improved attack against the RPM-based VC protocols for securely outsourcing large-scale matrix-related computations, which is further extended in the latter [15]. The key of the attack in [15] is to count the number of zero-valued elements in the RPM-based ciphertexts. Additionally, a strict proof is provided to show that all of the RPM-based VC protocols do not hold the privacy-preserving property under the attack in [15]. However, if two distinct inputs have the same number of zero-valued elements, an adversary cannot distinguish these two inputs by using the attack in [15]. In practice, users' possible matrices might usually contain the same number of zero-valued elements, making the attack in [15] disabled. For example, the matrices in the signal process always contain no zero-valued elements. This results in a limited application range of the attack in [15].

This paper continues the line of privacy research on the RPM-based protocols. We first dig out a natural question of RPM-based protocols; i.e., for an encrypted matrix based on the RPM, all elements in each row are masked by multiplying an identical random number, and all elements in each column are masked by dividing another identical random number, which provides a chance to crack the multiplier of each row and the divisor of each column by computing the greatest common division (GCM) of each row and the least common multiple (LCM) of each column. Based on this, we then develop a novel attack against the privacy-preserving property of the RPM-based VC protocols, which relies heavily on the GCM and LCM of each row and column in the encrypted matrices.

1.1. Related Work

More and more researchers pay attention to outsourcing computation due to its tremendous benefits. The design of outsourcing computation should achieve security, privacy, and efficiency. To meet these requirements, the most common solution is to implement outsourcing computation in a verifiable way. The initial works [16–18] concentrated on designing a general VC protocol for securely outsourcing any computation, which is based on fully homomorphic encryption (FHE). Although such solutions are attractive, they are far from the practice, which is because their operations are extremely complicated even in small cases [19]. To reduce the complexity, Ananth et al. [20] developed an improved general solution based on the one-way functions or the decisional Diffie–Hellman assumption, which, however, is still too expensive.

Thus, the research priority of the VC protocols is shifted to enhance their operational efficiency. A feasible solution is to design the VC protocols for specific purposes, in which heavy cryptographic operations can be avoided. Earlier works were conducted by Atallah et al. [5,21], which introduced a framework for resource-limited users to outsource numerical and scientific computations to a trusted cloud. However, these two protocols do not consider how to verify the correctness of the cloud computational results, making them impractical. Later, Atallah et al. [22,23] proposed another two protocols, which, however, are still unsatisfactory. The former is designed using the secret sharing technique, leading to a high communication cost; the latter relies on two untrusted clouds without colluding with each other. To address these problems, a large number of improved solutions were proposed. For example, the VC protocols for securely outsourcing linear programming [24,25], matrix inversion [6,11], matrix–matrix multiplication [7,26], matrix determinant [8], linear regression [27,28], the large-scale system of linear equations [10,29–35], compressed sensing reconstruction [12], and non-negative matrix factorization [36] have been investigated. Tang et al. [37] presented a methodology called PILE for privacy-preserving federated learning with verifiable perturbations. The study aims to address privacy concerns in federated learning by adding verifiable perturbations to the model updates. In [38], the authors proposed a method for achieving privacy-preserving and verifiable support vector machine (SVM) training in the cloud. The study addresses privacy and integrity concerns when training SVM models in a cloud environment. All of these improved protocols involve only basic algebra operations and thus are highly efficient.

Among the existing works, RPM is widely adopted in the design of VC protocols for securely outsourcing matrix-related computations [5–12]. In such protocols, cloud users can efficiently utilize RPM to encrypt a plaintext matrix and decrypt a ciphertext matrix. This implies that the input and the output of user privacy are both protected by RPM. For matrix inversion, matrix–matrix multiplication, and matrix decomposition, cloud users in [5–7,9,11,12] directly employed RPM to encrypt their matrices and decrypt the cloud computational results. For a matrix determinant, Lei et al. [8] combined RPM with the lower–upper (LU) decomposition to design an efficient VC protocol. For linear regression and large-scale systems of linear equations, data owners in [10] masked their coefficient matrices and reconstructed their actual solutions with the help of RPM. In these protocols, the most complicated operation is just 1-sparse matrix–matrix multiplication,

making them highly efficient. In addition, their cheating-resistant strategies can ensure that the probability that incorrect solutions are accepted by the clients is negligible.

Most notably, all of the above protocols build their privacy-preserving properties on RPM. RPM was once thought to be a decent technique to guarantee user privacy in VC protocols. Until the recent works [14,15], Zhao et al. proposed an attack against the privacy of RPM with respect to the density of encrypted matrices. The disadvantage of this attack is that the number of the zero-valued elements of users' possible inputs must be different, which is hard to guarantee in practice.

1.2. Contribution

In this paper, we focus on the privacy-preserving vulnerability in RPM and propose a novel attack against the RPM-based VC protocols. Our contributions are summarized as follows:

- We design an attack against the privacy-preserving vulnerability of RPM, which is based on the GCD and the LCM of each row and column in the encrypted matrices. It is the first time to allow an adversary to reconstruct the values of the inputs from the RPM-based ciphertexts. In addition, we take three representative RPM-based VC protocols as examples to illustrate how the proposed attack works. At last, we conduct the experiments to further validate the effectiveness of our attack.
- To formalize the privacy-preserving vulnerability of RPM under our attack, we present two novel definitions of the privacy-preserving property with respect to the GCD and the LCM of each row and column in the encrypted matrices, which are designed by using two indistinguishability experiments under chosen-plaintext attack (CPA) and ciphertext-only attack (COA). Then, we provide strict proof to show that an RMP-based VC protocol is not private-preserving.
- We present a detailed comparison between the proposed attack and the state-of-the-art [15]. The proposed attack allows an adversary to distinguish the encrypted matrices over two distinct inputs by decrypting the received ciphertext, instead of counting the number of zero-valued elements like the state of the art [15]. We also provide massive experimental results to further validate the advantages and disadvantages of the proposed attack and the attack in [15].

1.3. Organization

The rest of this paper is organized as follows: we introduce the preliminaries in Section 2. In Section 3, we present a framework of verifiable computation protocols and two novel definitions of the privacy-preserving property based on the CPA and the COA. In Section 4, we propose an attack against the privacy-preserving vulnerability of RPM, which is designed based on the GCD and the LCM of each row and column in the encrypted matrices. We also provide a comparison between the proposed attack and the attack in [15]. Section 5 conducts the experiments to evaluate the performance of the proposed attack and the attack in [15]. In Section 6, we present the conclusion of this paper.

2. Preliminaries

In this section, we first introduce the basic concepts for verifiable computation and then provide the details of random permutation masking, which is widely adopted in the design of VC protocols for secure outsourcing matrix-related computations.

2.1. Verifiable Computation

In this subsection, we focus on the framework of verifiable computation, which is depicted in Figure 1.

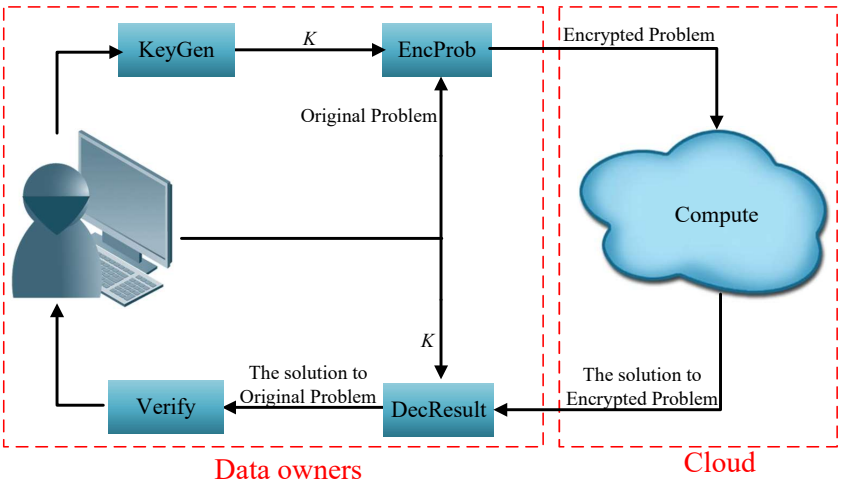

Figure 1. The system framework of a VC protocol for secure outsourcing computation.

Definition 1 (The framework of a VC protocol)**.** *There are five probabilistic polynomial-time (PPT) algorithms in a verifiable computation protocol, which is defined as follows...*

- $KeyGen(1^\lambda, \gamma)$ $\rightarrow K$: *Given a security parameter* λ *and a special parameter* γ *related to the input problem* Φ, *the client produces a system key* K.
- $EncProb(\Phi; K) \rightarrow \Phi'$: *Given a large-scale problem* Φ, *the user employs* K *to encrypt* Φ *into* Φ', *which is then sent to the cloud for the solution.*
- $Compute(\Phi') \rightarrow \sigma'$: *Once obtaining the encrypted problem* Φ', *the cloud computes its solution* σ', *which is associated with the solution to* Φ.
- $DecResult(\sigma'; K) \rightarrow \sigma$: *After receiving the masked solution* σ', *the client utilizes* K *to decrypt it into* σ.
- $Verify(\sigma; K) \rightarrow \perp$: *Upon the input of* σ, *the client verifies whether it is the correct solution to* Φ. *If yes, the client outputs* $\perp = 1$; *otherwise, outputs* $\perp = 0$.

2.2. Privacy-Preserving Property of a VC Protocol

In practice, the outsourced data usually contain user privacy, which might bring huge economic benefits. Thus, for the sake of interests, the cloud might try its best to reveal user privacy from the outsourced data. In the previous [5–12], it was claimed that a passive PPT adversary can hardly distinguish the outputs of EncProb over two distinct inputs. Based on the indistinguishability under chosen-plaintext attack (IND-CPA) and the indistinguishability under ciphertext-only attack (IND-COA), two formal definitions of privacy against passive adversary are formalized as follows:

Definition 2 (Privacy against the CPA [13,15])**.** *For a VC protocol, the following experiment implemented by a PPT adversary* $\mathcal{A}$ *is taken into account:*
Experiment $Exp_{\mathcal{A}}^{IND-CPA}[VC, \lambda, \gamma]$:

$$
\begin{aligned}
K &\leftarrow KeyGen(1^\lambda, \gamma) \\
(\Phi_0, \Phi_1) &\leftarrow \mathcal{A}^{PubEncProb_K(\cdot)}(1^\lambda, \gamma) \\
b &\xleftarrow{\$} \{0,1\} \\
\Phi'_b &\leftarrow EncProb(K, \Phi_b) \\
b' &\leftarrow \mathcal{A}^{PubEncProb_K(\cdot)}(\Phi_0, \Phi_1, \Phi'_b) \\
If\ b' &= b, outputs\, 1; else, outputs\, 0.
\end{aligned}
$$

where Φ'_b is called as the challenge ciphertext. The oracle $\mathsf{PubEncProb}_K(\Phi)$ asks $\mathsf{EncProb}(\Phi; K)$ to generate the encrypted value Φ'. Trivially, the output of $\mathsf{PubEncProb}_K(\cdot)$ is obviously probabilistic, which is fed back to the adversary $\mathcal{A}$. Using this experiment, the advantage of $\mathcal{A}$ is defined as follows:

$$Adv_{\mathcal{A}}^{IND-CPA}(VC, \lambda, \gamma) = \left| Pr\left[Exp_{\mathcal{A}}^{IND-CPA}[VC, \lambda, \gamma] = 1\right] - \frac{1}{2} \right|$$

If a VC protocol is IND-CPA private, it will satisfy that

$$Adv_{\mathcal{A}}^{IND-CPA}(VC, \lambda, \gamma) \leq negl(\lambda),$$

where $negl(\cdot)$ is a negligible function.

Definition 3 (Privacy against the COA [13,15])**.** *For a VC protocol, the following experiment executed by a PPT adversary $\mathcal{A}$ is taken into consideration:*
Experiment *$Exp_{\mathcal{A}}^{IND-COA}[VC, \lambda, \gamma]$:*

$$\begin{aligned} K &\leftarrow \mathsf{KeyGen}(1^\lambda, \gamma) \\ (\Phi_0, \Phi_1) &\leftarrow \mathcal{A}(1^\lambda, \gamma) \\ b &\xleftarrow{\$} \{0,1\} \\ \Phi'_b &\leftarrow \mathsf{EncProb}(K, \Phi_b) \\ b' &\leftarrow \mathcal{A}(\Phi'_b) \\ If\ b' &= b, outputs\, 1; else, outputs\, 0. \end{aligned}$$

The advantage of $\mathcal{A}$ in the above experiment is defined as follows:

$$Adv_{\mathcal{A}}^{IND-COA}(VC, \lambda, \gamma) = \left| Pr\left[Exp_{\mathcal{A}}^{IND-COA}[VC, \lambda, \gamma] = 1\right] - \frac{1}{2} \right|.$$

A VC protocol is IND-COA private only in the case of

$$Adv_{\mathcal{A}}^{IND-COA}(VC, \lambda, \gamma) \leq negl(\lambda).$$

2.3. Random Masking Permutation

In this subsection, we first describe the random permutation technique, and then introduce how to utilize this technique to mask an arbitrary matrix.

Generally, the random permutation technique can be expressed as

$$\tau = \begin{pmatrix} 1 & 2 & \ldots & n-1 & n \\ \omega_1 & \omega_2 & \ldots & \omega_{n-1} & \omega_n \end{pmatrix},$$

where $1 \leq \omega_i \leq n$, and $\omega_i \neq \omega_j$ for any $i \neq j$. For ease of description, let $\omega_i = \tau(i)$ and $i = \tau^{-1}(\omega_i)$, where τ and τ^{-1} represent a random permutation function and its inversion. Let $\delta_{x,y}$ denote the Kronecker delta function, i.e.,

$$\delta_{x,y} = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases}$$

RPM is based on two well-designed invertible matrices, $\mathbf{P}_1$ and $\mathbf{P}_2$, which are computed by

$$\begin{cases} \mathbf{P}(i_1, j_1) = \alpha_{i_1}\delta_{\tau_1(i_1),j_1}, & 1 \le i_1, j_1 \le m \\ \mathbf{Q}(i_2, j_2) = \beta_{i_2}\delta_{\tau_2(i_2),j_2}, & 1 \le i_2, j_2 \le n \end{cases} \tag{1}$$

Note that α_{i_1} and β_{i_2} are recommended to be random integers for the accuracy of outsourcing computation. In this paper, we suppose that α_{i_1} and β_{i_2} are randomly selected from $\{1, 2, \ldots, 2^\lambda - 1\}$, where λ is the security parameter of VC protocols. Then, we introduce RPM using the following theorem:

Theorem 1. *Given two invertible matrices,* $\mathbf{P}$ *and* $\mathbf{Q}$*, in (1), a large-scale matrix* $\mathbf{X}$ *can be efficiently encrypted into* $\mathbf{Y} = \mathbf{PXQ}^{-1}$ *by computing* $\mathbf{Y}(i, j) = \frac{\alpha_i}{\beta_j}\mathbf{X}(\tau_1(i), \tau_2(j))$*, where* $\mathbf{Y} = \mathbf{PXQ}^{-1}$ *is a secure encryption scheme with IND-CPA privacy.*

Proof. Let $Enc_K(\cdot)$ denote the encryption scheme $\mathbf{Y} = \mathbf{PXQ}^{-1}$, where $\mathbf{Y}(i, j) = \frac{\alpha_i}{\beta_j}\mathbf{X}(\tau_1(i), \tau_2(j))$, $1 \le i \le m$, and $1 \le j \le n$. The input, output, and key of $Enc_K(\cdot)$ are $\mathbf{X}$, $\mathbf{Y}$, and $\{\{\alpha_i\}_{1\le i\le m}, \{\beta_j\}_{1\le j\le n}, \tau_1, \tau_2\}$, respectively. First, we introduce the following security experiment about the IND-CPA privacy of $Enc_K(\cdot)$.

Experiment $\mathrm{Exp}_{\mathcal{A}}^{IND-CPA}[Enc_K, \lambda, \gamma]$:

$$\begin{aligned} K &\leftarrow \mathsf{KeyGen}(1^\lambda, m, n), \\ (X_0, X_1) &\leftarrow \mathcal{A}^{\mathsf{PubEnc}_K(\cdot)}(1^\lambda, m, n) \\ b &\xleftarrow{\$} \{0, 1\} \\ Y_b' &\leftarrow \mathsf{Enc}_K(X_b) \\ b' &\leftarrow \mathcal{A}^{\mathsf{PubEnc}_K(\cdot)}(X_0, X_1, Y_b') \\ &If\ b' = b,\ outputs\ 1;\ else,\ outputs\ 0. \end{aligned}$$

Based on the above experiment, the advantage of the adversary $\mathcal{A}$ can be defined as

$$\begin{aligned} &Adv_{\mathcal{A}}^{IND-CPA}(Enc_K, \lambda, \gamma) \\ &\quad = \left| Pr\left[Exp_{\mathcal{A}}^{IND-CPA}[Enc_K, \lambda, \gamma] = 1\right] - \frac{1}{2} \right| \end{aligned}$$

If $Enc_K(\cdot)$ is IND-CPA private, it will satisfy that

$$Adv_{\mathcal{A}}^{IND-CPA}(Enc_K, \lambda, \gamma) \le negl(\lambda).$$

Next, we show that $Enc_K(\cdot)$ achieves the property of IND-CPA privacy. Recall that $\{\alpha_i\}_{1\le i\le m}$ and $\{\beta_j\}_{1\le j\le n}$ are randomly selected with λ bits, and τ_1 and τ_2 are the random permutation of m and n elements. Thus, the expected time of brute-force attack on the key space to guess $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, $\{\beta_1, \beta_2, \ldots, \beta_n\}$, τ_1, and τ_2 is $O(2^\lambda)$, $O(2^\lambda)$, $O(m!)$, and $O(n!)$, respectively. For a large-scale matrix, its dimensions m and n must be sufficiently large. In such way, a PPT adversary can hardly crack $\{\alpha_i\}_{1\le i\le m}$, $\{\beta_j\}_{1\le j\le n}$, τ_1, and τ_2. This also means that the adversary can hardly distinguish two large-scale plaintext matrices, X_0 and X_1, from the ciphertext matrix Y_b' randomly generated by either of them. As a consequence, we can obtain that $Pr\left[Exp_{\mathcal{A}}^{IND-COA^{MUL}}[Enc_K, \lambda, m, n] = 1\right] \to \frac{1}{2}$. After that, we can dirctly conclude that $Enc_K(\cdot)$ is IND-CPA private from $Adv_{\mathcal{A}}^{IND-CPA}(Enc_K, \lambda, \gamma) \le negl(\lambda)$.

Another thing to note is that the time complexity of $Enc_K(\cdot)$ is $O(mn)$, which is substantially cheaper than performing matrix-related computations, such as matrix inversion, matrix determinant, and matrix–matrix multiplication. Thereafter, a large-scale matrix $\mathbf{X}$ can be efficiently encrypted into $\mathbf{Y}$ via $Enc_K(\cdot)$. □

2.4. Notations

In this subsection, we summarize the main notations adopted in this paper. Let $\mathbf{A}(i,*)$ denote the i-th row in the matrix $\mathbf{A}$, $\mathbf{A}(*,j)$ denote the j-th column in $\mathbf{A}$, $\mathbf{A}(i,j)$ or $a_{i,j}$ denote the element in the i-th row and the j-th column of $\mathbf{A}$, $\mathbf{A}^{-1}$ denote the inversion of $\mathbf{A}$, $|\mathbf{A}|$ denote the determinant of $\mathbf{A}$, $\mathbf{A}^{m\times n}$ denote an $m \times n$ matrix, $lcm(a,b)$ denote the least common multiple of a and b, and $gcd(a,b)$ denote the greatest common divisor of a and b.

3. Our Modified Formal Definitions

In the above section, there are three formal definitions of a VC protocol. The first is about the compact framework, and the last two are about the privacy-preserving property. These definitions aim to work as guidelines for designing a privacy-preserving VC protocol. In this section, we revised these three definitions due to the following reasons:

First, RPM-based VC protocols are mainly used for securely outsourcing matrix-related computations. According to Theorem 1, an input matrix $\mathbf{X}^{m\times n}$ is encrypted by the following two steps:

- *Step 1.* The position of each element in $\mathbf{X}$ is rearranged under two random permutation functions τ_1 and τ_2, i.e., $\mathbf{T}(i,j) = \mathbf{X}(\tau_1(i), \tau_2(j))$, where $1 \le i \le m$ and $1 \le j \le n$.
- *Step 2.* Each element in $\mathbf{T}$ is encrypted by multiplying a random number, i.e., $\mathbf{Y}(i,j) = \frac{\alpha_i}{\beta_j}\mathbf{T}(i,j)$.

In such way, the algorithms KeyGen and EncProb in the RPM-based VC can be further clarified. Thus, we introduce a novel framework to especially describe the RPM-based VC protocols.

Second, some inherent natures of RPM might result in the information leakage of the input matrices, which affects the privacy-preserving property in the RPM-based solutions. However, most of the existing protocols lack such privacy-preserving explorations. For example, to the best of our knowledge, there has been no study on how the greatest common divisor (GCD) and the least common multiple (LCM) of each row and column in the encrypted matrices affect the privacy-preserving property of RPM-based VC protocols. Thus, we redefine the privacy-preserving property with respect to the GCD and the LCM for RPM-based protocols.

3.1. RPM-Based Verifiable Computation

Definition 4 (The framework of an RPM-based VC protocol)**.** *Let Φ be a concrete matrix-related computation to be outsourced, whose inputs are supposed to be $\mathbf{X}_1^{m_1\times n_1}, \mathbf{X}_2^{m_2\times n_2}, \ldots,$ and $\mathbf{X}_L^{m_L\times n_L}$. For ease of description, the sizes of the input matrices are denoted by $\gamma = \{m_1 \times n_1, m_2 \times n_2, \ldots, m_L \times n_L\}$. The framework of an RMP-based VC protocol is presented as follows:*

- *KeyGen$(1^\lambda, \gamma) \to K$: With the input of a security parameter λ and a special parameter γ, the client generates a key $K = \{(\mathbf{P}_1, \mathbf{Q}_1), (\mathbf{P}_2, \mathbf{Q}_2), \ldots, (\mathbf{P}_L, \mathbf{Q}_L)\}$. For $1 \le l \le L$, $\mathbf{P}_l$ and $\mathbf{Q}_l$ are determined by (1).*
- *EncProb$(\Phi; K) \to \Phi'$: Given a large-scale problem Φ $(\mathbf{X}_1^{m_1\times n_1}, \mathbf{X}_2^{m_2\times n_2}, \ldots, \mathbf{X}_L^{m_L\times n_L})$ and the system key K, the client computes $\mathbf{Y}_l = \mathbf{P}_l\mathbf{X}_l\mathbf{Q}_l^{-1}$, where $1 \le l \le L$. The client outputs the encrypted problem $\Phi'(\mathbf{Y}_1^{m_1\times n_1}, \mathbf{Y}_2^{m_2\times n_2}, \ldots, \mathbf{Y}_L^{m_L\times n_L})$.*
- *Compute$(\Phi') \to \sigma'$: Upon the input of the encrypted problem Φ', the cloud calculates its solution σ'.*
- *DecResult$(\sigma'; K) \to \sigma$: Given the encoded solution σ', the client decrpyts it into σ.*
- *Verify$(\sigma; K) \to \perp$: Given the decoded solution σ, the client checks its correctness. If yes, the client outputs $\perp = 1$; otherwise, outputs $\perp = 0$.*

3.2. The Privacy-Preserving Property with Respect to the GCD and the LCM

An RPM-based VC protocol for securely outsourcing a matrix-related computation consists of the above five-tuple (KeyGen, EncProb, Compute, DecResult, Verify). All of the input matrices in the outsourced problem are encrypted based on RPM. Thus, we can

randomly pick one of the input matrices as an example to describe the privacy-preserving property. With this in mind, we formally define the privacy-preserving property with respect to the GCD and the LCM for RPM-based VC protocols. In our indistinguishability experiments, the adversary is denoted as $\mathcal{A}$, and an arbitrary input matrix of the original problem Φ is denoted as $\mathbf{X}$ with the size of $m \times n$. The details are given below.

According to Theorem 1, the encrypted matrix in the RPM-based VC protocol is computed by $\mathbf{Y}(i,j) = \frac{\alpha_i}{\beta_j}\mathbf{T}(i,j) = \frac{\alpha_i}{\beta_j}\mathbf{X}\left(\tau_1(i), \tau_2(j)\right)$, where $\mathbf{T}$ can be regarded as an intermediate transformation matrix. Thus, we can have the following two observations:

- Each row and column in $\mathbf{X}$ are first masked by rearranging their positions, i.e.,

$$\mathbf{T}(i,j) = \mathbf{X}(\tau_1(i), \tau_2(j)). \tag{2}$$

- Each row and column in $\mathbf{X}$ are further masked by multiplying a random number, i.e.,

$$\mathbf{Y}(i,j) = \frac{\alpha_i}{\beta_j}\mathbf{T}(i,j). \tag{3}$$

Combing the above observations, we can obtain

$$\begin{cases} \mathbf{Y}(i,*) = \alpha_i \mathbf{T}^{(1)}(i,*), & 1 \le i \le m \\ \mathbf{Y}(*,j) = \dfrac{1}{\beta_j}\mathbf{T}^{(2)}(*,j), & 1 \le j \le n \end{cases} \tag{4}$$

where $\mathbf{T}^{(1)}(i,j) = \frac{1}{\beta_j}\mathbf{T}(i,j)$ and $\mathbf{T}^{(2)}(i,j) = \alpha_i \mathbf{T}(i,j)$. It is straightforward that

- If $\mathbf{T}^{(1)}(i,*)$ is composed of two or more integers, these integral elements in $\mathbf{Y}(i,*)$ will have the common divisor α_i, making it possible to decode the secret α_i.
- If $\mathbf{T}^{(2)}(*,j)$ consists of two or more decimals, these nonintegral elements in $\mathbf{Y}(*,j)$ will have the common multiple β_j that makes them become integers at the same time, which provides a chance to decipher the secret β_j.

We can then observe that

- If β_j is prime to at least one entry in $\mathbf{T}^{(2)}(*,j)$, one can have $\beta_j = lcm(\beta'_{1,j}, \beta'_{2,j}, \ldots, \beta'_{m,j})$, where $\beta'_{i,j}$ is determined by (6).
- If α_i is prime to at least one entry in $\mathbf{T}(i,*)$, one can have $\alpha_i = gcd(\alpha'_{i,1}, \alpha'_{i,2}, \ldots, \alpha'_{i,n})$, where $\alpha'_{i,j} = \mathbf{T}^{(1)}(i,j)$ and $1 \le j \le n$.

Please refer to the next Section 4.1 for the detailed proof.

After cracking all the values of $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and $\{\beta_1, \beta_2, \ldots, \beta_n\}$, an adversary is able to decode the intermediate matrix $\mathbf{T}$ according to (3). This implies that the values of the original matrix $\mathbf{X}$ will be exposed to the adversary. In such a way, the adversary can easily distinguish the encrypted matrix over two distinct inputs.

Definition 5 (Privacy with respect to the GCD and the LCM against the CPA)**.** *For an RPM-based VC protocol, an experiment associated with a PPT adversary $\mathcal{A}$ is taken into consideration:*

Experiment $Exp_{\mathcal{A}}^{IND-CPA^{MUL}}[VC, \lambda, m, n]$:

$$\begin{aligned}
K &\leftarrow \mathit{KeyGen}(1^{\lambda}, m, n)\\
(\mathbf{X}_0, \mathbf{X}_1) &\leftarrow \mathcal{A}^{\mathit{PubEncProb}_K(\cdot)}(1^{\lambda}, m, n)\\
b &\xleftarrow{\$} \{0,1\}\\
\mathbf{Y}_b &\leftarrow \mathit{EncProb}(K, \mathbf{X}_b)\\
\bar{\beta}_j &\leftarrow \mathcal{A}^{lcm(\beta'_{1,j}, \beta'_{2,j}, \ldots, \beta'_{m,j})}, \textit{ where } 1 \le j \le n\\
\bar{\alpha}_i &\leftarrow \mathcal{A}^{gcd(\alpha'_{i,1}, \alpha'_{i,2}, \ldots, \alpha'_{i,n})}, \textit{ where } 1 \le i \le m\\
\mathbf{X}'_b &\leftarrow \mathcal{A}^{scale(\mathbf{Y}_b; \bar{\alpha}_{1,j}, \bar{\alpha}_{2,j}, \ldots, \bar{\alpha}_{m,j}, \bar{\beta}_{1,j}, \bar{\beta}_{2,j}, \ldots, \bar{\beta}_{m,j};)}\\
b' &\leftarrow \mathcal{A}^{\mathit{PubEncProb}_K(\cdot)}(\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}'_b)\\
&If\ b' = b,\ outputs\, 1;\ else,\ outputs\, 0.
\end{aligned}$$

From Definition 5, we can easily obtain the definition of privacy with respect to the GCD and the LCM against the COA, which is as follows:

Definition 6 (Privacy with respect to the GCD and the LCM against the COA)**.** *For an RPM-based VC protocol, an experiment associated with a PPT adversary $\mathcal{A}$ is taken into account:*
Experiment $Exp_{\mathcal{A}}^{IND-COA^{MUL}}[VC, \lambda, m, n]$:

$$\begin{aligned}
K &\leftarrow \mathit{KeyGen}(1^{\lambda}, m, n)\\
(\mathbf{X}_0, \mathbf{X}_1) &\leftarrow \mathcal{A}^{PubEncProb_K(\cdot)}(1^{\lambda}, m, n)\\
b &\xleftarrow{\$} \{0,1\}\\
\mathbf{Y}_b &\leftarrow \mathit{EncProb}(K, \mathbf{X}_b)\\
\bar{\beta}_j &\leftarrow \mathcal{A}^{lcm(\beta'_{1,j}, \beta'_{2,j}, \ldots, \beta'_{m,j})}, \textit{ where } 1 \le j \le n\\
\bar{\alpha}_i &\leftarrow \mathcal{A}^{gcd(\alpha'_{i,1}, \alpha'_{i,2}, \ldots, \alpha'_{i,n})}, \textit{ where } 1 \le i \le m\\
\mathbf{X}'_b &\leftarrow \mathcal{A}^{scale(\mathbf{Y}_b; \bar{\alpha}_{1,j}, \bar{\alpha}_{2,j}, \ldots, \bar{\alpha}_{m,j}, \bar{\beta}_{1,j}, \bar{\beta}_{2,j}, \ldots, \bar{\beta}_{m,j};)}\\
b' &\leftarrow \mathcal{A}(\mathbf{X}'_b)\\
&If\ b' = b,\ outputs\, 1;\ else,\ outputs\, 0.
\end{aligned}$$

With the help of the above experiment, we define the advantage of $\mathcal{A}$ as

$$\begin{aligned}
&Adv_{\mathcal{A}}^{IND-COA^{MUL}}(VC, \lambda, m, n)\\
&= \left| Pr\left[Exp_{\mathcal{A}}^{IND-COA^{MUL}}[VC, \lambda, m, n] = 1\right] - \frac{1}{2} \right|.
\end{aligned} \tag{5}$$

If an RPM-based VC protocol is IND-COA private with respect to the GCD and the LCM, it will achieve

$$Adv_{\mathcal{A}}^{IND-COA^{MUL}}(VC, \lambda, m, n) \le negl(\lambda),$$

where $negl(\cdot)$ is a negligible function.

Note that (1) from Definitions 5 and 6, one can observe that in order to distinguish the input matrices, the adversary in Definition 5 is allowed to access an oracle $PubEncProb_K(\cdot)$, while the adversary in Definition 6 has no oracle to access. This implies that the adversary in the IND-CPA has a more powerful capability compared with the adversary in the IND-COA. Consequently, an RPM-based VC protocol that holds IND-CPA privacy with respect to the GCD and the LCM must also hold IND-COA privacy with respect to the GCD and

the LCM. In other words, if it does not hold IND-COA privacy with respect to the GCD and the LCM, it will also not hold IND-CPA privacy with respect to the GCD and the LCM. (2) If the input matrices of the original problem are all revealed by the adversary, its solution will also be disclosed.

4. Privacy-Preserving Vulnerability of Random Permutation Masking

In this section, we first reveal the privacy-preserving vulnerability of RPM-based VC protocols according to Definition 6, and then provide a comparison between the proposed attack and the attack in [15]. At last, we take three representative RPM-based VC protocols as examples to show how our attack works.

4.1. Privacy-Preserving Vulnerability

According to Definition 6, the key to breaking the privacy-preserving property is to crack $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and $\{\beta_1, \beta_2, \ldots, \beta_n\}$ by using the GCD and the LCM. For solving $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, each row in $\mathbf{Y}$ should contain multiple integers. However, the division performed on each column of $\mathbf{T}$ in (3) enables that the entries in each row of $\mathbf{Y}$ might be all decimals. In addition, in order to solve $\{\beta_1, \beta_2, \ldots, \beta_n\}$, it is required that each column in $\mathbf{Y}$ has multiple decimals, which is contrary to the requirement of solving $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Thus, we should first determine $\{\beta_1, \beta_2, \ldots, \beta_n\}$ so that all entries in $\mathbf{Y}$ can be transformed into integers, and then $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ are also able to be solved. As a result, the key in our attack is to prove that each column in $\mathbf{Y}$ contains one or more decimals. Before our proof, we first introduce a helpful lemma.

Lemma 1. *If an integer η is sufficiently large, the number of its factors is much smaller than $\frac{\eta}{2}$.*

Proof. For any integer η, we can observe that its factors from largest to smallest are η, $\frac{\eta}{2}$ if $mod(\eta, 2) = 0$, $\frac{\eta}{3}$ if $mod(\eta, 3) = 0, \ldots$, and 1 in turn. If η is sufficiently large, $(\frac{\eta}{2} - \frac{\eta}{3})$ is much greater than 1. Thus, the number of the factors of η is much smaller than $\frac{\eta}{2}$. □

Theorem 2. *In an RPM-based VC protocol, the probability that each column in $\mathbf{Y}$ has at least one decimal is close to 1.*

Proof. From (4), the i-th entry in $\mathbf{Y}(*, j)$ is an integer only if β_j is the factor of $\mathbf{T}^{(2)}(i, j)$. According to Lemma 1, the probability that β_j is the factor of $\mathbf{T}^{(2)}(i, j)$ is much smaller than $\frac{1}{2}$. To enable that $\mathbf{Y}(*, j)$ is only composed of integers, β_j is required to be the factor of all entries in $\mathbf{T}^{(2)}(i, j)$. Thus, the probability that each column in $\mathbf{Y}$ has at least one decimal is much greater than $1 - (\frac{1}{2})^m$. In a VC protocol, the original matrix $\mathbf{X}$ usually has a large-scale size, and then $1 - (\frac{1}{2})^m$ is close to 1. Thus, we can conclude Theorem 2. □

In addition, if two input matrices have different numbers of zero-valued elements, the adversary can easily distinguish them when either of their RPM-based ciphertexts is given. Specifically, the adversary observes these two input matrices whose number of zero-valued elements is the same as that of the given RPM-based encrypted matrix and determines who is the right input matrix. This is always in effect due to the fact that RPM does not change the number of zero-valued elements of the input matrices. Based on the above discussion, we summarize our attack in Algorithm 1. One thing to note is that $\mathbf{X}'_b$ in step 12 of Algorithm 1 is equal to $\mathbf{T}$ with a high probability, which is formalized in Theorem 3. To prove Theorem 3, we first introduce two useful lemmas.

Lemma 2. *In Algorithm 1, the probability of $\beta_j = \bar{\beta}_j$ can be represented as $1 - (1 - P)^m$, where $1 \leq j \leq n$ and P is the probability that any two integers are relatively prime.*

Proof. From the second equation in (4), it can be observed that $\beta'_{i,j}$ in step 4 of Algorithm 1 satisfies

$$\beta'_{i,j} = \frac{\beta_j}{\mathcal{K}_{\beta_{i,j}}}, \tag{6}$$

where $\mathcal{K}_{\beta_{i,j}}$ is determined by the following two steps:

- Step 1. Initializing $\mathcal{K}_{\beta_{i,j}} = 1$ and $\mathcal{T}_{\beta_{i,j}} = \mathbf{T}^{(2)}(i,j)$.
- Step 2. Repeatedly updating $\mathcal{K}_{\beta_{i,j}} = \mathcal{K}_{\beta_{i,j}} \times gcd(\beta_j, \mathcal{T}_{\beta_{i,j}})$ and $\mathcal{T}_{\beta_{i,j}} = \frac{\mathcal{T}_{\beta_{i,j}}}{gcd(\beta_j, \mathcal{T}_{\beta_{i,j}})}$ until $gcd(\beta_j, \mathcal{T}_{\beta_{i,j}})$ is equal to 1.

Note that $\beta'_{i,j}$ is equal to β_j in the case of $gcd(\beta_j, \mathbf{T}^{(2)}(i,j)) = 1$. Thereafter, if β_j is prime to at least one entry in $\mathbf{T}^{(2)}(*,j)$, one can have

$$\bar{\beta}_j = lcm(\beta'_{1,j}, \beta'_{2,j}, \ldots, \beta'_{m,j}) = \beta_j.$$

Next, we discuss the probability that at least one entry in $\mathbf{T}^{(2)}(*,j)$ is prime to β_j. Since there are m entries in $\mathbf{T}^{(2)}(*,j)$, the probability that β_j and at least one entry in $\mathbf{T}^{(2)}(*,j)$ are coprime can be computed as $1-(1-P)^m$, where P is the probability that any two integers are relatively prime. With the increase in m, one can find that $1-(1-P)^m$ is convergent to 1. Thus, Lemma 2 can be concluded. □

Algorithm 1 The attack based on the LCM and the GCD

Require: Two distinct input matrices $\mathbf{X}_0$ and $\mathbf{X}_1$, the encrypted matrix $\mathbf{Y}$, where $\mathbf{Y}(i,j) = \frac{\alpha_i}{\beta_j}\mathbf{X}_b(\tau_1(i), \tau_2(j))$, $b \xleftarrow{\$} \{0,1\}$, $1 \le i \le m$ and $1 \le j \le n$.

Ensure: b'.

1: **if** $\mathbf{X}_0$ and $\mathbf{X}_1$ have the same number of zero-valued elements, **then**
2: **for** $j = 1 : n$ **do**
3: **for** $i = 1 : m$ **do**
4: The adversary calculates $\beta'_{i,j}$, which is the least multiple that makes $\mathbf{Y}(i,j)$ integer.
5: **end for**
6: The adversary obtains $\bar{\beta}_j = lcm(\beta'_{1,j}, \beta'_{2,j}, \ldots, \beta'_{m,j})$, which is the least common multiple that makes all of the entries in $\mathbf{Y}(*,j)$ integers.
7: **end for**
8: The adversary computes $\mathbf{T}^{(1)} = \mathbf{Y} \times diag(\bar{\beta}_1, \bar{\beta}_2, \ldots, \bar{\beta}_n)$.
9: **for** $i = 1 : m$ **do**
10: The adversary calculates the greatest common divisor $\bar{\alpha}_i$ of all the entries in $\mathbf{Y}(i,*)$.
11: **end for**
12: The adversary computes $\mathbf{X}'_b = diag(\frac{1}{\bar{\alpha}_1}, \frac{1}{\bar{\alpha}_2}, \ldots, \frac{1}{\bar{\alpha}_m}) \times \mathbf{T}^{(1)}$.
13: **if** $\mathbf{X}'_b$ and $\mathbf{X}_0$ have the same set of matrix elements,
14: The adversary outputs $b' = 0$;
15: **else if** $\mathbf{X}'_b$ and $\mathbf{X}_1$ have the same set of matrix elements,
16: The adversary outputs $b' = 1$;
17: **else**
18: The adversary outputs $b' \xleftarrow{\$} \{0,1\}$.
19: **end if**
20: **else**
21: **if** $\mathbf{X}_b$ and $\mathbf{X}_0$ have the same number of zero-valued elements,
22: The adversary outputs $b' = 0$;
23: **else if** $\mathbf{X}_b$ and $\mathbf{X}_1$ have the same number of zero-valued elements,
24: The adversary outputs $b' = 1$;
25: **else**
26: The adversary outputs $b' \xleftarrow{\$} \{0,1\}$.
27: **end if**

Lemma 3. *In Algorithm 1, the probability of $\alpha_i = \bar{\alpha}_i$ can be represented as $1-(1-P)^n$, where $1 \leq i \leq m$ and P is the probability that any two integers are relatively prime.*

Proof. According to the first equation in (4), if α_i is prime to at least one entry in $\mathbf{T}(i,*)$, one can have

$$\bar{\alpha}_i = gcd(\mathbf{T}^{(1)}(i,1), \mathbf{T}^{(2)}(i,2), \ldots, \mathbf{T}^{(2)}(i,n)) = \alpha_i. \tag{7}$$

The probability that α_i is prime to at least one entry in $\mathbf{T}^{(1)}(i,*)$ can be represented as $1-(1-P)^n$, which is convergent to 1 with the growth of n. Thus, we can obtain Lemma 3. □

Theorem 3. *In Algorithm 1, if the numbers of columns and rows in $\mathbf{X}$ are both sufficiently large, the probability that $\mathbf{X}'_b$ is equal to $\mathbf{T}$ is close to 1.*

Proof. According to Algorithm 1, $\mathbf{X}'_b$ is equal to $\mathbf{T}$ only if $\bar{\alpha}_i = \alpha_i$ and $\bar{\beta}_j = \beta_j$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. From Lemma 2 and Lemma 3, for $1 \leq i \leq m$ and $1 \leq j \leq n$, the probability that $\bar{\alpha}_i = \alpha_i$ and $\bar{\beta}_j = \beta_j$ can be computed as $(1-(1-P)^m)^n(1-(1-P)^n)^m$. According to the binomial theorem [39], if $|x| < 1$, then

$$(1+x)^n = 1 + \sum_{k=1}^{\infty} C_n^k x^k,$$

where $C_n^k = \frac{n(n-1)\cdots(n-k+1)}{k!}$ and $C_n^0 = 1$. Thus, one can obtain

$$(1-(1-P)^m)^n = 1 + \sum_{k=1}^{\infty} C_n^k (1-P)^{mk}.$$

Due to $0 < P < 1$, one can compute $\lim_{m,n\to\infty} n^k(1-P)^{mk} = 0$ in the case that m and n are both approaching infinity simultaneously and there is no order in which one approaches infinity before the other. Then, one can have

$$\lim_{m,n\to\infty}(1-(1-P)^m)^n = 1. \tag{8}$$

In a similar way, one can also have

$$\lim_{m,n\to\infty}(1-(1-P)^n)^m = 1. \tag{9}$$

Combining (8) and (9), we can conclude that

$$\lim_{m,n\to\infty}(1-(1-P)^m)^n(1-(1-P)^n)^m = 1.$$

Thereafter, if m and n are large enough, the probability of $\bar{\alpha}_i = \alpha_i$ and $\bar{\beta}_j = \beta_j$ is convergent to 1, where $1 \leq i \leq m$ and $1 \leq j \leq n$. Thus, we can conclude Theorem 3. □

Theorem 4. *An RPM-based VC protocol is not IND-COA private with respect to the GCD and the LCM of each row and column in the encrypted matrices.*

Proof. In a VC protocol, the outsourced matrix $\mathbf{X}$ must be so large-scale that the resource-limited client cannot perform any complex computation on $\mathbf{X}$ locally. Otherwise, there is no need for the client to resort to a powerful cloud. In such way, according to Theorem 3, we can obtain that $\mathbf{X}'_b(i,j) = \mathbf{T}(i,j) = \mathbf{X}(\pi_1(i), \pi_2(j))$ with a high probability. This implies that an adversary can reconstruct the values of all entries in the original matrix. Although the order of the elements in each row and column of the original matrix is still unknown, the adversary can distinguish two distinct input matrices by comparing the values of matrix elements. For example, the adversary can determine the original matrix by judging which input matrix is equal to the decoded matrix after simply sorting the matrix entries

from smallest to largest. This makes us conclude that $b = b'$ with a high probability and $Pr\left[Exp_{\mathcal{A}}^{IND-COA^{MUL}}[VC, \lambda, m, n] = 1\right] \rightarrow 1$. According to (5), we can then have

$$Adv_{\mathcal{A}}^{IND-COA^{MUL}}(VC, \lambda, m, n) \rightarrow \frac{1}{2} \nleq negl(\lambda).$$

Thereafter, RMP-based VC protocols are not IND-COA with respect to the GCD and the LCM of each row and column in the encrypted matrices. □

Furthermore, the proposed attack can also work in the case that the inputs of the RPM are nonintegral. The reason is as follows: For an arbitrary decimal $\hat{x}_{i,j} \in \mathbf{X}$, it can be transformed into $\hat{x}_{i,j} = \frac{x_{i,j}}{10^{e_j}}$, where e_j is the maximal number of decimal digits among the elements in the j-th column, making $x_{i,j}$ an integer. Under this circumstance, the RPM-based encrypted matrix can be transformed into $\mathbf{Y}(i,j) = \frac{\alpha_i}{10^{e_j}\beta_j}\mathbf{T}(i,j)$. This also means that the proposed attack can be utilized to crack the values of α_i and $10^{e_j}\beta_j$, where $1 \leq i \leq m$ and $1 \leq j \leq n$.

4.2. Compared with the State of the Art [15]

In the previous works [15], Zhao et al. also presented an attack against the RPM-based VC protocols, which are based on the matrix density. Specifically, in their privacy-preserving experiments with respect to the matrix density, two plaintext matrices, $\mathbf{X}_0$ and $\mathbf{X}_1$, are required to have different numbers of zero-valued elements. The adversary distinguishes these two input matrices by counting the number of zero-valued elements in the encrypted matrix $\mathbf{Y}$. However, there are two limitations: (1) The attack in [15] is disabled when $\mathbf{X}_0$ and $\mathbf{X}_1$ have the same number of zero-valued elements. The reason is that RPM does not change the number of zero-valued elements in $\mathbf{X}_0$ and $\mathbf{X}_1$. (2) The attack in [15] cannot be used to reconstruct the values of the input matrix from the encrypted matrix.

The proposed attack is built on the LCM and the GCD of each row and column in the encrypted matrices and thus can avoid the limitations in the previous [15].

4.3. Further Discussion

We further analyze the privacy-preserving vulnerability in the RPM-based VC protocols for securely outsourcing matrix inversion [11], matrix determinant [8], and matrix-matrix multiplication [7], which are of different types.

- To solve the inversion of the large-scale matrix $\mathbf{X}$, the client first utilizes Theorem 1 to encrypt $\mathbf{X}$ into $\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$ in [11]. Once receiving $\mathbf{Y}$, the cloud invokes Algorithm 1 to estimate $\mathbf{T}$, and then computes $\mathbf{T}^{-1}$. Due to $\mathbf{T}(i,j) = \mathbf{X}(\pi_1(i), \pi_2(j))$ and $\mathbf{T}^{-1}(i,j) = \mathbf{X}^{-1}(\pi_2(i), \pi_1(j))$, the cloud can recover the values of all entries in $\mathbf{X}$ and $\mathbf{X}^{-1}$.
- To compute the determinant of a large-scale matrix $\mathbf{X}$, the client first employs Theorem 1 to obtain $\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1}$ in [8]. After receiving $\mathbf{Y}$, the cloud executes Algorithm 1 to estimate $\mathbf{T}$, and then computes $|\mathbf{T}|$. Since $\mathbf{T}(i,j) = \mathbf{X}(\pi_1(i), \pi_2(j))$, the cloud can obtain the values of all entries in $\mathbf{X}$ and the exact value of $|\mathbf{X}|$.
- To determine the large-scale matrix–matrix multiplication $\mathbf{Z}_x = \mathbf{X}_1\mathbf{X}_2$, the client first uses Theorem 1 to calculate $\mathbf{Y}_1 = \mathbf{P}_1\mathbf{X}_1\mathbf{P}_3^{-1}$ and $\mathbf{Y}_2 = \mathbf{P}_3\mathbf{X}_2\mathbf{P}_2^{-1}$ in [7]. While obtaining $\mathbf{Y}_1$ and $\mathbf{Y}_2$, the cloud runs Algorithm 1 to estimate $\mathbf{T}_1$ and $\mathbf{T}_2$, and then calculates $\mathbf{Z}_t = \mathbf{T}_1\mathbf{T}_2$. Because of $\mathbf{T}_1(i,j) = \mathbf{X}_1(\pi_1(i), \pi_2(j))$, $\mathbf{T}_2(i,j) = \mathbf{X}_2(\pi_1(i), \pi_2(j))$, and $\mathbf{Z}_t = \mathbf{Z}_x(\pi_1(i), \pi_2(j))$, the cloud can reconstruct the values of all entries in $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_1\mathbf{X}_2$.

5. Performance Evaluation

In this section, we conduct massive experiments to evaluate the performance of the proposed attack, which is implemented by using Matlab 2018a. The workstation is built on Intel(R) Core(TM) i5-7200U CPU and 8 GB RAM. We also provide a comparison of the

proposed attack and the previous attack in [15]. Each experimental result is an average of 1000 trials.

5.1. The Proposed Attack

To check the effectiveness of the proposed attack, we introduce an indicator *successful probability*, which represents the probability that an adversary successfully decodes the values of all entries in the original matrices. In the experiment, every element in the original matrices is in the range of 0 to 255, which is common in image processing. The *successful probabilities* of the proposed attack against RPM are presented in Figure 2.

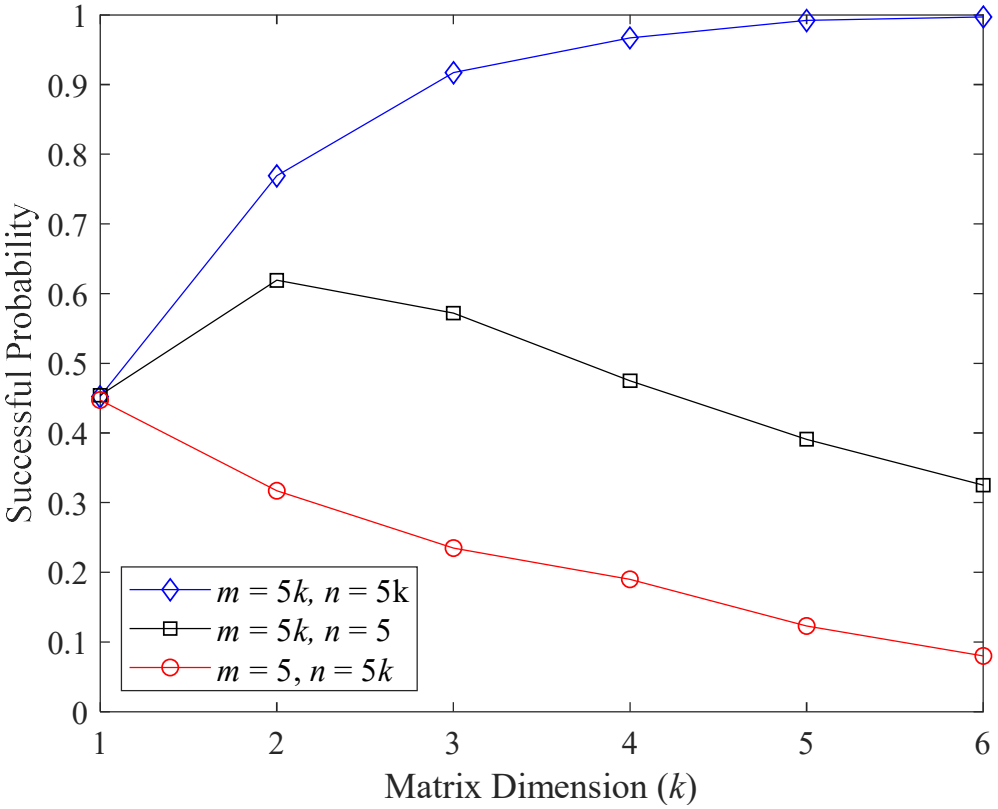

Figure 2. The successful probabilities of the proposed attack against RPM.

From Figure 2, we can have the following observations: (1) the *successful probability* increases with the dimension of the original matrix; (2) the *successful probability* is near to 1 in the case of 30×30 input matrices, which would be much closer to 1 with the input of larger matrices according to Theorem 3; and (3) given a certain row number, the *successful probability* decreases with the column number in the large region, and vice versa. The reason is as follows: according to Theorem 3, the *successful probability* of the proposed attack is determined by $(1-(1-P)^n)^m(1-(1-P)^m)^n$, which converges to 0 as m or n approaches to infinity; (4) the *successful probability* of the fixed row number m is always lower than that of the fixed column number n with the same value. The reason is that the proposed attack first decodes $\{\beta_1, \beta_2, \ldots, \beta_n\}$, and then eliminates their effect on cracking $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. Thus, the decryption of $\{\beta_1, \beta_2, \ldots, \beta_n\}$ is more error-prone than that of $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. According to Lemma 2, the probability of successfully decoding $\{\beta_1, \beta_2, \ldots, \beta_n\}$ can be represented as $(1-(1-P)^m)^n$, which is proportional to m and is inversely proportional to n. Thereafter, the *successful probability* of a given column number n degrades faster than that of a given row number m with the same value.

5.2. The Comparison of the Different Attacks against RPM

We compare the proposed attack with the attacks in [13,15] by computing

$$Pr\left[Exp_{\mathcal{A}}^{IND-COA}[VC, \lambda, \gamma] = 1\right], \tag{10}$$

i.e., the advantage that the adversary $\mathcal{A}$ provides a correct answer, which is summarized in Table 1. In the experiments, an adversary randomly chooses one plaintext matrix from $\{\mathbf{X}_0, \mathbf{X}_1\}$ as the input of RPM. For ease of description, we define two types of matrix sets: $SM \subset \mathbf{Z}^{m \times n}$ and $DM \subset \mathbf{Z}^{m \times n}$, where $\mathbf{X} \in SM$ is an $(\frac{m}{2}, \frac{n}{2})$-sparse matrix element and $\mathbf{X} \in DM$ is a dense matrix. Note that in the $(\frac{m}{2}, \frac{n}{2})$-sparse matrix, the numbers of zero-valued elements in each row and column are $\frac{m}{2}$ and $\frac{n}{2}$, respectively.

Table 1. The comparsion of the advantages $Pr\left[Exp_{\mathcal{A}}^{IND-COA}[VC, \lambda, \gamma] = 1\right]$ on two different attacks.

Case	Description		Matrix Dimension ($m \times n$)					
			6×10	12×20	18×30	24×40	30×50	36×60
1	$\mathbf{X}_0 \in SM, \mathbf{X}_1 \in DM$	The attack in [13]	1	1	1	1	1	1
		The attack in [15]	1	1	1	1	1	1
		The proposed attack	1	1	1	1	1	1
2	$\mathbf{X}_0 \in SM, \mathbf{X}_1 \in SM$	The attack in [13]	0.50	0.51	0.51	0.49	0.50	0.51
		The attack in [15]	0.49	0.51	0.50	0.49	0.51	0.51
		The proposed attack	0.65	0.90	0.96	0.99	0.99	1
3	$\mathbf{X}_0 \in DM, \mathbf{X}_1 \in DM$	The attack in [13]	0.51	0.50	0.49	0.51	0.51	0.51
		The attack in [15]	0.51	0.49	0.50	0.51	0.51	0.50
		The proposed attack	0.86	0.95	0.99	1	1	1

From Table 1, we can have the following observations: (1) As shown in Case 1, these three attacks can always distinguish the two input matrices with the different number of zero-valued elements. This is because that the RPM-based encryption does not change the number of zero-valued elements in the input matrices. (2) The attack in [13] in Cases 2 and 3 is always disabled, because the elements in the encrypted matrices based on RPM do not have the equal ratio relationship. (3) The attack in [15] is always disabled if the input matrices have the same number of zero-valued elements, which is provided in Cases 2 and 3. (4) According to Cases 1, 2, and 3, the proposed attack can differentiate two distinct input matrices, no matter whether they have the same zero-valued elements or not. The main reason is that the proposed attack depends mainly on the GCD and the LCM of each row and column in the encrypted matrices in addition to the number of zero-valued elements in the encrypted matrices. (5) The proposed attack has a smaller advantage in Case 2 than in Case 3. This is because a sparse matrix provides fewer integers for determining the masking values $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and $\{\beta_1, \beta_2, \ldots, \beta_n\}$. To sum up, the proposed attack has superiority over the state of the art [15].

To sum up, the attack proposed in this study offers a wider range of application scenarios compared with the current state-of-the-art approach [15]. This can be attributed to the following reasons: First, the protocol proposed in this research enables the decryption of the values of elements within the plaintext matrices. Unlike existing methods, which are limited to specific cases, this protocol allows for a more comprehensive cracking process. Second, the proposed protocol is not constrained by the number of zero elements present in the plaintext matrices. This flexibility distinguishes it from previous approaches and makes it suitable for a broader set of scenarios. By leveraging these advantages, the proposed attack demonstrates its potential for enhancing the applicability and effectiveness of decryption techniques in various contexts.

The aforementioned advantages make the proposed attack a highly viable option for researchers and practitioners alike. Its ability to crack the values of the elements in plaintext matrices, coupled with its unrestricted nature towards the number of zero elements, imbues the attack with remarkable adaptability and practicality. As a result, it is well suited for implementation in diverse scenarios, thereby contributing to the advancement of the field. Further investigation into the proposed attack's performance in real-world scenarios and its potential vulnerabilities is warranted to fully assess its strengths and weaknesses.

6. Conclusions

In this paper, we delved deeper into the privacy vulnerabilities associated with RPM. To begin, we introduce a privacy-preserving definition of RPM-based protocols by employing two indistinguishability experiments based on the chosen plaintext attack (CPA) and the chosen plaintext and ciphertext attack (COA). Subsequently, our research focuses on unveiling a novel attack against RPM that capitalizes on the indispensability of the greatest common divisor (GCD) and the least common multiple (LCM) values of encrypted matrices' rows and columns.

To illustrate the lack of privacy preservation within the RPM framework, we furnish comprehensive evidence along with detailed formal proofs. Our elucidation demonstrates that RPM is incapable of upholding the desired privacy preservation in the face of the proposed attack. Notably, in contrast with the state-of-the-art research [15], the proposed attack boasts several advantageous characteristics that warrant examination: First, our attack empowers adversaries to distinguish between encrypted matrices associated with two distinct inputs, without any requirement for the inputs to possess the dissimilar number of zero-valued elements. This capability exceeds the existing approaches in terms of discriminating power. Second, our attack enables adversaries to decode the plaintext matrix values from their RPM-based ciphertext counterparts. This breakthrough finding affirms the vulnerability of RPM to plaintext recovery attacks, which can compromise the confidentiality of sensitive information.

To validate the efficacy and implications of our proposed attack, we conducted an extensive series of experiments. The experimental results serve to corroborate the advantages identified earlier, shedding light on the potential risks inherent in using RPM-based protocols for privacy-preserving computations. By shedding light on the privacy vulnerabilities of RPM and showcasing the practicality of our attack method, this paper highlights the importance of strengthening existing privacy mechanisms and developing robust countermeasures. Future research efforts should focus on addressing the identified vulnerabilities to safeguard the privacy and confidentiality of sensitive information processed using RPM-based protocols. In future work, we will try to crack the RPM-encrypted matrix in a deterministic way; i.e., both the positions and values of matrix elements can be deterministically reconstructed.

Author Contributions: Conceptualization, Y.Y. and G.S.; methodology, Y.Y. and G.S.; software, Y.Y.; validation, Y.Y. and G.S.; formal analysis, Y.Y.; investigation, Y.Y.; resources, G.S.; data curation, G.S.; writing—original draft preparation, Y.Y. and G.S.; writing—review and editing, Y.Y. and G.S.; visualization, Y.Y.; supervision, Y.Y.; project administration, Y.Y.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Humanities and Social Sciences Research Project of the Chinese Ministry of Education under Grant No. 22YJCZH217 and the Graduate Education Reform Project of the Zhongnan University of Economics and Law under Grant No. YJ20230043.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors thank the anonymous referees and editors for their valuable suggestions and comments to improve this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, Y.H.; Tan, T.; Zhu, Y. Face identification based on singular value decomposition and data fusion. *Chin. J. Comput.-Chin. Ed.* **2000**, *23*, 649–653.
2. Murphy, K.P. *Machine Learning: A Probabilistic Perspective*; MIT Press: Cambridge, MA, USA, 2012.
3. Al-Dhuraibi, Y.; Paraiso, F.; Djarallah, N.; Merle, P. Elasticity in Cloud Computing: State of the Art and Research Challenges. *IEEE Trans. Serv. Comput.* **2018**, *11*, 430–447. [CrossRef]
4. Gennaro, R.; Gentry, C.; Parno, B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Proceedings of the Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010; Proceedings 30; Springer: Berlin/Heidelberg, Germany, 2010; pp. 465–482.
5. Atallah, M.J.; Pantazopoulos, K.N.; Rice, J.R.; Spafford, E.E. Secure outsourcing of scientific computations. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 2002; Volume 54, pp. 215–272.
6. Lei, X.; Liao, X.; Huang, T.; Li, H.; Hu, C. Outsourcing Large Matrix Inversion Computation to A Public Cloud. *IEEE Trans. Cloud Comput.* **2013**, *1*, 1. [CrossRef]
7. Lei, X.; Liao, X.; Huang, T.; Heriniaina, F. Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. *Inf. Sci.* **2014**, *280*, 205–217. [CrossRef]
8. Lei, X.; Liao, X.; Huang, T.; Li, H. Cloud Computing Service: The Caseof Large Matrix Determinant Computation. *IEEE Trans. Serv. Comput.* **2015**, *8*, 688–700. [CrossRef]
9. Zhou, L.; Li, C. Outsourcing Eigen-Decomposition and Singular Value Decomposition of Large Matrix to a Public Cloud. *IEEE Access* **2016**, *4*, 869–879. [CrossRef]
10. Yu, Y.; Luo, Y.; Wang, D.; Fu, S.; Xu, M. Efficient, secure and non-iterative outsourcing of large-scale systems of linear equations. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6. [CrossRef]
11. Hu, C.; Alhothaily, A.; Alrawais, A.; Cheng, X.; Sturtivant, C.; Liu, H. A secure and verifiable outsourcing scheme for matrix inverse computation. In Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9. [CrossRef]
12. Zhang, Y.; Xiang, Y.; Zhang, L.Y.; Yang, L.X.; Zhou, J. Efficiently and securely outsourcing compressed sensing reconstruction to a cloud. *Inf. Sci.* **2019**, *496*, 150–160. [CrossRef]
13. Zhao, L.; Chen, L. A Linear Distinguisher and its Application for Analyzing Privacy-Preserving Transformation Used in Verifiable (Outsourced) Computation. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Republic of Korea, 4–8 June 2018; pp. 253–260.
14. Zhao, L.; Chen, L. On the Privacy of Matrix Masking-Based Verifiable (Outsourced) Computation. *IEEE Trans. Cloud Comput.* **2020**, *8*, 1296–1298. [CrossRef]
15. Zhao, L.; Chen, L. Sparse Matrix Masking-Based Non-Interactive Verifiable (Outsourced) Computation, Revisited. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 1188–1206. [CrossRef]
16. Chung, K.M.; Kalai, Y.; Vadhan, S. Improved delegation of computation using fully homomorphic encryption. In Proceedings of the Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010; Proceedings 30; Springer: Berlin/Heidelberg, Germany, 2010; pp. 483–501.
17. Barbosa, M.; Farshim, P. Delegatable homomorphic encryption with applications to secure outsourcing of computation. In Proceedings of the Topics in Cryptology–CT-RSA 2012: The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, 27 February– 2 March 2012; Proceedings; Springer: Berlin/Heidelberg, Germany, 2012; pp. 296–312.
18. Kalai, Y.T.; Raz, R.; Rothblum, R.D. How to delegate computations: The power of no-signaling proofs. In Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 31 May–3 June 2014; pp. 485–494.
19. Parno, B.; Howell, J.; Gentry, C.; Raykova, M. Pinocchio: Nearly Practical Verifiable Computation. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19–22 May 2013; pp. 238–252. [CrossRef]
20. Ananth, P.; Chandran, N.; Goyal, V.; Kanukurthi, B.; Ostrovsky, R. Achieving privacy in verifiable computation with multiple servers–without FHE and without pre-processing. In Proceedings of the International Workshop on Public Key Cryptography, Buenos Aires, Argentina, 26–28 March 2014; pp. 149–166.
21. Atallah, M.J.; Li, J. Secure outsourcing of sequence comparisons. *Int. J. Inf. Secur.* **2005**, *4*, 277–287. [CrossRef]
22. Benjamin, D.; Atallah, M.J. Private and Cheating-Free Outsourcing of Algebraic Computations. In Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust, Fredericton, NB, USA, 1–3 October 2008; pp. 240–245. [CrossRef]
23. Atallah, M.J.; Frikken, K.B. Securely outsourcing linear algebra computations. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 13–16 April 2010; pp. 48–59.
24. Wang, C.; Ren, K.; Wang, J. Secure and practical outsourcing of linear programming in cloud computing. In Proceedings of the 2011 Proceedings IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 820–828. [CrossRef]
25. Chen, F.; Xiang, T.; Yang, Y. Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud. *J. Parallel Distrib. Comput.* **2014**, *74*, 2141–2151. [CrossRef]
26. Zhang, X.; Jiang, T.; Li, K.C.; Castiglione, A.; Chen, X. New publicly verifiable computation for batch matrix multiplication. *Inf. Sci.* **2019**, *479*, 664–678. [CrossRef]

27. Chen, F.; Xiang, T.; Lei, X.; Chen, J. Highly Efficient Linear Regression Outsourcing to a Cloud. *IEEE Trans. Cloud Comput.* **2014**, *2*, 499–508. [CrossRef]
28. Yang, Y.; Xiong, P.; Huang, Q.; Chen, F. Secure and efficient outsourcing computation on large-scale linear regressions. *Inf. Sci.* **2020**, *522*, 134–147. [CrossRef]
29. Wang, C.; Ren, K.; Wang, J.; Wang, Q. Harnessing the Cloud for Securely Outsourcing Large-Scale Systems of Linear Equations. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 1172–1181. [CrossRef]
30. Chen, X.; Huang, X.; Li, J.; Ma, J.; Lou, W.; Wong, D.S. New Algorithms for Secure Outsourcing of Large-Scale Systems of Linear Equations. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 69–78. [CrossRef]
31. Salinas, S.; Luo, C.; Chen, X.; Li, P. Efficient secure outsourcing of large-scale linear systems of equations. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 1035–1043. [CrossRef]
32. Li, D.; Dong, X.; Cao, Z.; Wang, H. Privacy-preserving large-scale systems of linear equations in outsourcing storage and computation. *Sci. China Inf. Sci.* **2018**, *61*, 1–9. [CrossRef] [PubMed]
33. Salinas, S.; Luo, C.; Chen, X.; Liao, W.; Li, P. Efficient Secure Outsourcing of Large-Scale Sparse Linear Systems of Equations. *IEEE Trans. Big Data* **2018**, *4*, 26–39. [CrossRef]
34. Zhou, L.; Zhu, Y.; Choo, K.K.R. Efficiently and securely harnessing cloud to solve linear regression and other matrix operations. *Future Gener. Comput. Syst.* **2018**, *81*, 404–413. [CrossRef]
35. Ding, Q.; Weng, G.; Zhao, G.; Hu, C. Efficient and Secure Outsourcing of Large-Scale Linear System of Equations. *IEEE Trans. Cloud Comput.* **2021**, *9*, 587–597. [CrossRef]
36. Duan, J.; Zhou, J.; Li, Y. Secure and Verifiable Outsourcing of Large-Scale Nonnegative Matrix Factorization (NMF). *IEEE Trans. Serv. Comput.* **2021**, *14*, 1940–1953. [CrossRef]
37. Tang, X.; Shen, M.; Li, Q.; Zhu, L.; Xue, T.; Qu, Q. PILE: Robust Privacy-Preserving Federated Learning via Verifiable Perturbations. *IEEE Trans. Dependable Secur. Comput.* **2023**, 1–18. [CrossRef]
38. Hu, C.; Zhang, C.; Lei, D.; Wu, T.; Liu, X.; Zhu, L. Achieving Privacy-Preserving and Verifiable Support Vector Machine Training in the Cloud. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 3476–3491. [CrossRef]
39. Taylor, J. Work out pure mathematics A-level, by Betty Haines and Roger Haines. Pp 246.£ 7· 50. 1991. ISBN 0-333-54385-8 (Macmillan). *Math. Gaz.* **1991**, *75*, 469. [CrossRef]