Being Transparent is Merely the Beginning: Enforcing Purpose Limitation with Polynomial Approximation

Shuofeng Liu[1] Zihan Wang[1] Minhui Xue[2] Long Wang[3] Yuanchao Zhang[3] Guangdong Bai[1]

[1]*The University of Queensland* [2]*CSIRO's Data61* [3]*Information Security Department, Ant Finance*

Abstract

Obtaining the authorization of users (i.e., *data owners*) prior to data collection has become commonplace for online service providers (i.e., *data processors*), in light of the stringent data regulations around the world. However, it remains a challenge to uphold the principle of *purpose limitation*, which mandates that collected data should only be processed for the purpose that the data owner has originally authorized. In this work, we advocate *algorithm specificity*, as a means to enforce the purpose limitation principle. We propose ALGOSPEC, which obscures data to restrict its usability solely to an authorized algorithm or algorithm group. ALGOSPEC exploits the nature of *polynomial approximation* that given the input data and the highest order, any algorithm can be approximated with a unique polynomial. It converts the original authorized algorithm (or a part of it) into a polynomial and then creates a list of alternatives to the original data. To assess the efficacy and efficiency of ALGOSPEC, we apply it to the entropy method and Naive Bayes classification under datasets of different magnitudes from 10^2 to 10^6. ALGOSPEC significantly outperforms cryptographic solutions such as fully homomorphic encryption (FHE) in efficiency. On accuracy, it achieves a negligible Mean Squared Error (MSE) of 0.289 in the entropy method against computation over plaintext data, and identical accuracy (92.11%) and similar F1 score (87.67%) in the Naive Bayes classification.

1 Introduction

In recent years, there has been a notable increase in the stringency of data regulations worldwide, such as European Union General Data Protection Regulation (GDPR) [11] and California Consumer Privacy Act (CCPA) [8]. This proactive stance on user data regulation is a response to the growing concerns surrounding user data privacy and security. As a direct consequence, most online service providers (i.e., *data controllers* or *data processors*[1]) are mandated to obtain explicit consent from users (i.e., *data owners*) before initiating data collection. This approach ensures a heightened degree of transparency and accountability during data collection. For instance, Meta now requests explicit user consent before using their personal data for targeted advertising or content personalization [15]. Similarly, Amazon seeks permission from users to collect and store their browsing and purchase history for personalized recommendations [6]. All these shifts underscore the global commitment to safeguarding user privacy and granting data owners control over their personal data.

Despite the commendable strides made in ensuring transparency of data collection, the implementation of another crucial data regulation principle, i.e., the *purpose limitation principle* [13], continues to present challenges. This principle is another cornerstone in data protection, which stipulates that collected data must be utilized *exclusively* for the purpose that the data owner has originally authorized. This is particularly significant in the context of online data processing, which is the backbone of contemporary online services. Two typical scenarios are listed below.

Scenario #1. An online service provider collects user data through its mobile or web client, with its intended purpose disclosed in the associated privacy policy document. The client-side data processing is relatively transparent as it is more or less auditable (e.g., by reverse engineering the client-side software). A noticeable shift occurs once the data reaches the backend, where the data processing becomes nontransparent from the users. This places the users in an unfair position, as they lose visibility and control over how their data is handled.

Scenario #2. A financier collects personal data from individuals who apply for a loan. As part of its evaluation process, it engages a *third-party* credit assessment agency, e.g., Equifax, to assess the credit of the applicants. While the financier can certainly be transparent about its data collection and sharing practices, it still enforces its trust in the credit assessor onto the applicants.

Some existing techniques, such as FHE [32, 51, 54] and differential privacy (DP) [30, 34, 49], can be adapted for purpose limitation. They may fall short in some specific contexts

[1]Both are referred to as *data processors* without causing ambiguity.

though. FHE is effective at safeguarding the original data, but cannot prevent the data processor from performing unauthorized algorithms on the encrypted data. In certain situations when the processor possesses the encryption key (e.g., in ***Scenario #1***), it is feasible for them to reveal the results of unauthorized algorithms. On the other hand, DP enhances privacy by adding perturbations to the original data, but the perturbations may inadvertently compromise the accuracy of the intended algorithms.

Our work. In this work, we propose a novel approach named ALGOSPEC (standing for *algorithm specificity*), which obfuscates the data *exclusively* for the intended algorithm. ALGOSPEC targets to achieve two goals, i.e., algorithm specificity (***Goal #1***) and confidentiality of the original data (***Goal #2***). First, only the intended algorithm can lead to correct results, and no meaningful results can be obtained by unauthorized algorithms. Second, it is infeasible for an adversary operating in polynomial time to recover the original data from the obfuscated version.

The core technique to construct ALGOSPEC is to fit the entire or the core of the intended algorithm using *polynomial approximation*, leveraging its inherent feature that given the input data and the highest order, any algorithm can be accurately approximated using a distinctive polynomial representation (***Goal #1***). ALGOSPEC determines the highest order n based on the discrete logarithm problem, and then the coefficients and the constant term of the polynomial are uniquely determined. The polynomial is combined with the original data to obtain the obscured list M that can hide n and the original data (***Goal #2***) by blurring the non-constant and constant terms of the polynomial through predefined *patterns*.

We conduct comprehensive theoretical and empirical studies on ALGOSPEC to investigate its security, accuracy, and scalability. We prove that n is confidential, M is indistinguishable and finally the original data d is unrecoverable using code-based game-playing with the discrete logarithm problem and the decisional Diffie-Hellman problem. Based on that, the two desired goals of ALGOSPEC can be reduced to the indistinguishability of the original data against the chosen-plaintext attack.

We then experimentally benchmark ALGOSPEC against two typical data protection approaches, i.e., FHE and DP, on two widely-used algorithms in sensitive statistical analysis, i.e., the entropy method and Naive Bayes classification. The results show that ALGOSPEC significantly outperforms FHE in terms of efficiency, with nearly 13 times faster on the entropy method and 4 times faster on Naive Bayes classification. For the accuracy evaluation, we set a baseline by performing direct computation on the original data. ALGOSPEC can achieve an MSE of 0.289 compared with the baseline on entropy method, and identical test accuracy and similar F1 to the baseline on Naive Bayes classification. These findings highlight ALGOSPEC's effectiveness in preserving data privacy without compromising the data analysis. To further evaluate the scalability, we assess it with large-scale datasets with sizes up to 10^6. Its performance remains stable even as the dataset size increases, demonstrating that it is a practical and reliable data protection approach for real-world applications.

Contributions. Our main contributions can be summarized as follows.

- **Purpose limitation beyond legal regulation.** We emphasize the pressing necessity of practical mechanisms that go beyond mere legal manners to ensure purpose limitation, thereby safeguarding individuals' privacy rights and mitigating data misuse or abuse. For data owners, it is imperative that they possess *genuine* control over how their data is to be used. For data processors, they can retain verifiable evidence to demonstrate that the data is processed exclusively for the stated purposes.
- **A practical solution for purpose limitation.** We propose ALGOSPEC, a novel, efficient, and accurate method specifically designed for purpose limitation. It leverages polynomial approximation to fit the intended algorithm into a polynomial, and the original data then can be secured with polynomial-specific computation patterns.
- **A comprehensive study and evaluation.** We conduct a theoretical study on ALGOSPEC to formally prove its security against polynomial time attacks. We also conduct comprehensive experiments to study its performance, in terms of accuracy, efficiency, and scalability, demonstrating its practical applicability.

2 Problem Formulation

2.1 System settings

We define three participants in our problem settings, i.e., the user as the *data owner*, the *data processor*, and the *third party*, representing most of real-world scenarios in online services. The data processor collects user data, and determines and discloses the purpose of processing user data. It seeks to demonstrate to the user that their original data has been used *only* for the disclosed purpose. The data processor may involve a third party to perform (part of) the data processing. The data has to be obfuscated to accommodate *only* the disclosed purpose, before it is outsourced to the third party. We further elaborate and generalize these two scenarios as illustrated in Figure 1.

Scenario #1. In this scenario (Figure 1.(a)), the data processor consists of the client and the server. Data collection is performed by the client, where the intended purpose disclosed, e.g., through the associated privacy policy document. As the client runs on the user device, client-side data processing is transparent and auditable. In contrast, the server-side processing is nontransparent to the user, once the collected data reaches server side. Since the client and server both belong to the data processor, the algorithm is known to the client.

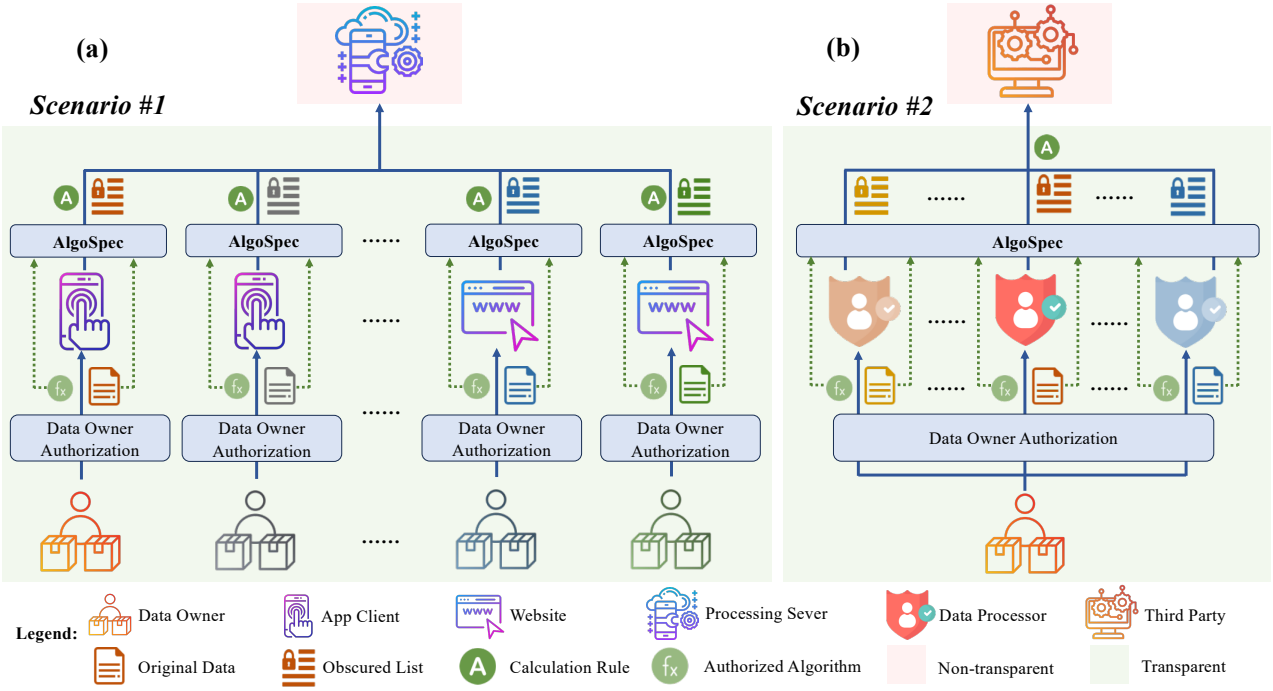


Figure 1: Two typical application scenarios of ALGOSPEC.

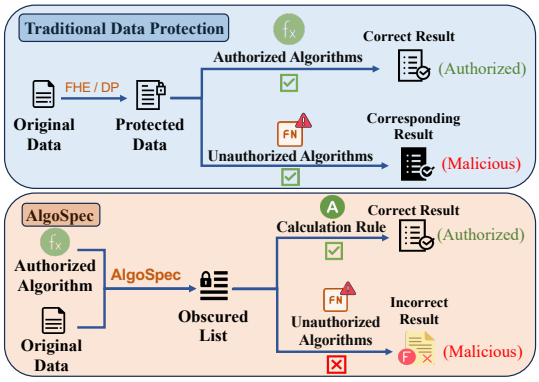


Figure 2: A comparison between the work models of ALGOSPEC and FHE/DP-based approaches.

Scenario #2. In this scenario (Figure 1.(b)), the data processor collects the original data and involves a third party for data processing. Recalling the financier and credit assessor example, the third party (i.e., credit assessor) owns more data than the individual data processor, e.g., the transaction history of a single user among multiple financiers.

Work model of ALGOSPEC. To position ALGOSPEC, we show the difference between its work model and FHE/DP-based approaches in Figure 2, explaining how it achieves the algorithm specificity. It combines the original data with authorized algorithms during the obfuscation. This ensures that the original data can be efficiently computed and analyzed exclusively under the umbrella of authorized algorithms. In contrast, FHE/DP-based approaches focus solely on modifying the original data, such that the resulting data can be processed under any algorithm.

2.2 Goals and Threat Model

ALGOSPEC is designed to align with the purpose limitation principle, which is commonly incorporated by major data regulations around the world, such as GDPR (Article 5, Chapter 2; Article 18, Chapter 3) [12], CCPA (1798.110; 1798.121) [8], The Privacy Act (Content 3, Chapter 6) [14]. In particular, we define two main security goals of ALGOSPEC below.

Goal #1. Algorithm specificity. This is to ensure that only executing the authorized algorithms can obtain correct results.

Goal #2. One-way obfuscation. This is to ensure the polynomial time adversary (PTA) cannot recover the original data.

Consistent with the setup of other related studies [17, 27], we assume the PTA as the attacker. The server of data processor in ***Scenario #1*** or the third party in ***Scenario #2*** is an honest-but-curious attacker, which attempts to recover the original data or perform unauthorized algorithms on the obtained obfuscated data. The attacker has both authorized and unauthorized algorithms.

3 Approach

In this section, we delve into the internals of ALGOSPEC. Figure 3 illustrates its general workflow, and the detailed

Table 1: Key notations used in this paper

Notations	Descriptions
$\mathbb{D}$	original dataset
$\mathbb{D}_f^{(s)}$	value of $\mathbb{D}$ at s^{th} row, f^{th} column
ℓ	unspecified value in dataset
Fx	authorized algorithm
F	polynomial approximation of Fx
n	highest order of F
Fn	malicious algorithms
M	obscured list
N	the number of elements in M
A	calculation rule
d	exponent adjustment factor
λ	coefficient adjustment factor
γ	the number of distances
Γ	distance list
r	result by ALGOSPEC
R	result by applying Fx on $\mathbb{D}$
G	cyclic group
$<\mathcal{G}>$	cyclic group generated by $\mathcal{G}$
$\mathcal{G}$	cyclic group generator
g	random generator
$\mathcal{P}$	possibility

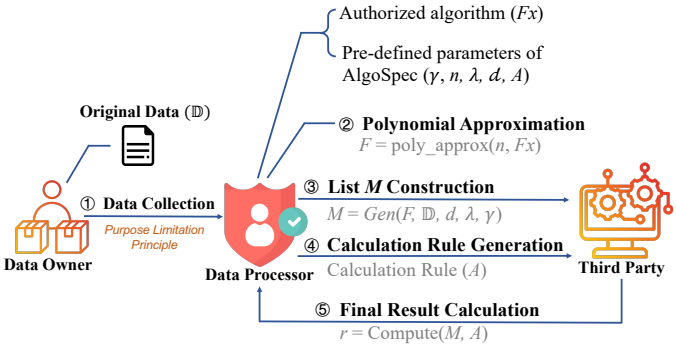


Figure 3: Overall workflow of ALGOSPEC.

algorithm is presented in Appendix A.1. Table 2 provides a step-by-step procedure of using ALGOSPEC. In the remaining of this section, we explain the key techniques of ALGOSPEC. To facilitate understanding, Table 1 lists key notations used.

3.1 Polynomial Approximation

A polynomial consists of variables (also known as indeterminates) and coefficients, combined using addition, multiplication, and non-negative integer exponents [38]. In its most general form, a polynomial can be expressed as Eq. 1.

$$P(x) = \alpha_n x^n + \alpha_{n-1} x^{n-1} + \ldots + \alpha_2 x^2 + \alpha_1 x + \beta, \quad (1)$$

where x is the variable, $\alpha_i x^i$s are the non-constant terms, α_is are coefficients, β is the constant term, and the non-negative integer n represents the highest order of x in the polynomial, which is called the degree of the polynomial.

Polynomial approximation has been proposed to represent a complex function with a polynomial using the target dataset

Table 2: Step-by-step Procedure of ALGOSPEC (***Scenario #2***)

Setup (Data Owner)
• Data owner authorizes data processor to collect data with the desired purpose disclosed.
Setup (Data Processor)
• Data processor obtains authorization for data collection, and discloses the desired algorithm Fx to be applied to the collected data.
• Data processor collects original data from data owner.
• Data processor determines n of the polynomial.
• Data processor converts Fx to polynomial F by polynomial approximation.
• Data processor acquires several components from ALGOSPEC, including d (exponent adjustment factor), λ (coefficient adjustment factor), γ (the number of distances), *pairs* (the pairs selected following distances), and *times* (times that pairs are used).
Setup (Third Party)
• Third party sets up execution environment.
Processing
• ALGOSPEC generates obscured list M using $\mathbb{D}$, F, d, λ, γ.
• When M is generated, calculation rule A is generated automatically.
• Data processor sends M and A to third party.
• Third party computes the result r through M and A.
• Third party sends r to data processor.

[31]. It is based on the foundation established by Mhaskar et al. [41] that the fitted polynomial can be used as an alternative to the original algorithm to obtain almost identical results on the given dataset. This ensures that ALGOSPEC sacrifices minimal precision loss during the obfuscation.

ALGOSPEC adopts polynomial fitting [48], a typical machine learning technique, for the approximation. It defines the range of original data and proceeds to uniformly select a specific number of data points within this range. These selected data points are then processed using the authorized algorithm, producing corresponding output values. The input-output pairs are then used for a polynomial fitting with Mean Squared Error as the loss function, given a specific degree n (determination of n is discussed soon). This is essentially a machine learning process, and it ends up with a polynomial with its uniquely determined coefficients and the constant term. ALGOSPEC works on the entire algorithm in general, but data processors can select core components of the algorithm to approximate. For example, some simple operations like normalization and addition can be excluded during the approximation to retain efficiency without losing accuracy.

The algorithm specificity (***Goal #1***) of ALGOSPEC stems from the *uniqueness* of the polynomial with respect to the degrees used for polynomial approximation. Funaro et al. [31] prove that, given an algorithm, the approximation with different degrees leads to different coefficients and the constant term. Once a polynomial has been derived to represent the authorized algorithm based on the given degree n, computing this polynomial will only yield the same result as applying

the authorized algorithm to the original data.

Determining and hiding n. Revealing n could enable the attacker to fit a polynomial of the authorized algorithm, potentially allowing them to recover the original data from the final result (i.e., r in Figure 3) by solving a high-order equation. In Section 5.3, we present an analysis on the complexity of this attack. By default, ALGOSPEC still keeps n confidential. It leverages the decisional Diffie-Hellman problem (DDH) to generate n (detailed in Section A.2), and constructs the obscured list M to hide it (detailed soon in Section 3.2).

3.2 Obscured List Construction

Based on the polynomial derived, ALGOSPEC proceeds to create an obscured list M, designed to conceal n. The idea is to break down the calculation of the polynomial into a large number ($\gg n$) of basic units. This involves dealing separately with the non-constant terms (Section 3.2.1), i.e., $\alpha_i x^i$ in Eq. 1, and the constant term (Section 3.2.2), i.e., β, capitalizing on the polynomial's inherent properties.

3.2.1 Processing of non-constant terms $\alpha_i x^i$

ALGOSPEC transforms those non-constant terms into a list M', which contains $2n$ elements denoted as m spanning from m_1 to m_{2n}. The intuition behind this is to split the sensitive value into segments, such that the value can be revealed only when merging all segments. To this end, ALGOSPEC introduces two essential factors for each term $\alpha_i x^i$, i.e., a coefficient adjustment factor λ_i, and an exponent adjustment factor d_i, which adjust the coefficient and the exponent, respectively. They compose two vectors of size n, which are denoted as λ and d. Below we explain how each m in M' is constructed.

For m_1 in M', ALGOSPEC sets $m_1 \leftarrow \frac{\alpha_n}{\lambda_n} x^{n-d_n}$, meaning that the first non-constant term depends on λ_n and d_n. Specifically, we divide the coefficient α_n by the corresponding adjustment factor λ_n, and subtract the order n of x from the value of the corresponding position d_n in the exponent adjustment vector. For m_2 in M', we set $m_2 \leftarrow \lambda_n x^{d_n}$ with the same λ_n and d_n as what have been used in constructing m_1. By doing so, the first non-constant term of the polynomial $\alpha_n x^n$ can be obtained by simply multiplying m_1 with m_2. We therefore organize m_1 and m_2 as a pair (m_1, m_2). This process is repeated for the remaining m_i, till the last pair (m_{2n-1}, m_{2n}) where we set $m_{2n-1} \leftarrow \frac{\alpha_1}{\lambda_1} x^{1-d_1}$ and $m_{2n} \leftarrow \lambda_1 x^{d_1}$. Consequently, we have the list M' containing $2n$ ms, i.e., n pairs.

After constructing the list M', ALGOSPEC obtains the way of calculating the result of the polynomial without the constant term β. By computing $m_1 \times m_2 + ... + m_{2n-1} \times m_{2n}$, the result being equal to the calculation of $\alpha_n x^n + \alpha_{n-1} x^{n-1} + ... + \alpha_1 x$ can be derived, i.e.,

$$m_1 \times m_2 + ... + m_{2n-1} \times m_{2n} = \alpha_n x^n + ... + \alpha_1 x. \quad (2)$$

The two vectors λ and d mainly serve to construct the two factors of a term, while also introducing randomness into the distribution of m_is. An extremely large or small λ may lead to precision loss in calculations, so we constrain $\lambda_i \in (0, 1)$ following Gaussian distribution in ALGOSPEC's implementation. We also set $d_i \in [1, 9] \cap \mathbb{Z}$, ensuring efficient computation of m_{2i}. They are adjustable in the actual deployment of ALGOSPEC.

Next, ALGOSPEC provides the way of processing the constant term β, to obtain the completed obscured list M.

3.2.2 Processing of constant term β

ALGOSPEC decomposes β into another list B which has the same pair-wise format as M'. B is of size $N - 2n$ ($N \gg 2n$ and N is an even number), such that the obscured list M of size N can be assembled by concatenating B with M'. We denote the elements in B from m_{2n+1} to m_N. Similar to M', when we design B, we guarantee that computing $m_{2n+1} \times m_{2n+2} + ... + m_{N-1} \times m_N$ achieves the result being equal to β as Eq. 3, i.e.,

$$m_{2n+1} \times m_{2n+2} + ... + m_{N-1} \times m_N = \beta. \quad (3)$$

To ensure that the final computation result matches the desired β, ALGOSPEC utilizes the last two ms to make necessary adjustments. Thus, the initial step involves the random generation of $(N - 2n - 2)$ ms in the list B. We construct the pairs in B to be indistinguishable from those in M', so that the adversary cannot learn n from their distribution patterns defined below.

Definition 1. *(Pattern). A pattern refers to the distribution of the distance* $\lfloor \log \frac{m_i}{m_{i+1}} \rfloor$*, where* i *is odd and* m_i*s are the elements in* M'.

ALGOSPEC follows five steps to construct B.

Step 1: ALGOSPEC creates adequate new distances[2] and puts them into the list Γ. The number of distances in Γ is denoted as τ. Following each distance in Γ, it builds a list Θ containing τ pairs, i.e., (θ_1, θ_2),..., $(\theta_{2\tau-1}, \theta_{2\tau})$.

Step 2: ALGOSPEC finds the distance with the highest number of occurrences from M', and selects a random number K greater than this maximal occurrence. Then, by leveraging Γ, it adds new distances such that all distances end up with an occurrence number close to K. To achieve this, $(m_{2n+1}, m_{2n+2}), \cdots, (m_{N-3}, m_{N-2})$ is generated, where each m_i follows Gaussian distribution, i.e., $m_i \sim \mathcal{N}(0, \theta_i)$.

Step 3: ALGOSPEC constructs m_{N-1} in B by selecting a pair, such as (θ_i, θ_{i+1}), from Θ and makes $m_{N-1} \sim \mathcal{N}(0, \theta_i)$.

Step 4: For m_N, ALGOSPEC needs a value to ensure that Eq. 3 can be satisfied. Therefore, it sets m_N as

$$m_N = \frac{\beta - (m_{2n+1} \times m_{2n+2} + ... + m_{N-3} \times m_{N-2})}{m_{N-1}}. \quad (4)$$

[2]The number of new distances should not be less than ρ which is the number of distance within n pairs, and we prove its security in Section A.2.

Step 5: ALGOSPEC shuffles all the tuples (m_{2n+1}, m_{2n+2}), ..., (m_{N-1}, m_N), and then reassigns the elements in all tuples to B following the new order.

The generation of list B is a direct outcome of the progression from Step 1 to Step 5. By combining Eq. 2 and Eq. 3, we can derive a new equation

$$\begin{aligned} &m_1 \times m_2 + \ldots + m_{2n-1} \times m_{2n} + \ldots + m_{N-1} \times m_N \\ &= \alpha_n x^n + \ldots + \beta. \end{aligned} \tag{5}$$

During this entire process, the calculation rule A is automatically formulated as follows.

Calculation rule A. The third party can compute the final result using M and A. Specifically, they apply the calculation rule in Eq. 6 to obtain the final result r of the authorized algorithm. The value of r should closely resemble the actual result R obtained when the original authorized algorithm is executed directly, i.e., $r = R + \eta$, where η is a negligible value signifying the margin of error.

$$r = m_1 \times m_2 + \ldots + m_{2n-1} \times m_{2n} + \ldots + m_{N-1} \times m_N. \tag{6}$$

Here, the data processor can delegate part of the complex calculation of M to the third party. Specifically, for those m_is where i is odd, which entail high exponent calculations, the processor can select a relatively small k_i and compute $x_i^{k_i}$ locally, denoted as X_i. Then it can request the third party to compute $X_i^{(i-d_i)/k_i}$ to construct m_i. The third party is not able to recover the original x_i in an efficient way from X_i and $(i-d_i)/k_i$, guaranteed by the discrete logarithm problem [46].

4 Case Studies

We demonstrate the application of ALGOSPEC with two widely-used algorithms, i.e., the entropy method and the Naive Bayes classification. We select them in our study primarily due to the diverse mathematical operations involved in their computation steps. Beyond primitive arithmetic operations, they encompass operations such as logarithmic operations, floating point operations, and probabilistic calculations. These operations have proven challenging for existing cryptographic algorithms designed to operate on ciphertexts [18].

4.1 Entropy Method

The entropy method (EM) is a multi-criteria decision analysis commonly used in measuring value dispersion in many practical cases [23]. The detailed operation of the EM can be summarized by the following steps.

$$x_{ij}^{'} = \frac{x_{ij} - \min(x_{ij})}{\max(x_j) - \min(x_j)}, \tag{7}$$

$$p_{ij} = \frac{x_{ij}^{'}}{\sum_{i=1}^{m} x_{ij}^{'}}, \tag{8}$$

$$e_j = -\frac{1}{\ln(n)} \sum_{i=1}^{m} p_{ij} \ln(p_{ij}), \tag{9}$$

$$W_j = -\frac{1 - e_j}{\sum_{j=1}^{n}(1 - e_j)}, \tag{10}$$

where i represents the i^{th} sample, j represents the j^{th} feature and m represents the number of samples.

Entropy method with ALGOSPEC. Data processor firstly normalizes the data using Z-Score based on the Eq. 7, and then computes p for each value using Eq. 8. Next, according to Eq. 9, data processor applies polynomial approximation on $p_{ij}\ln(p_{ij})$ in order to generate obscured lists from M_1 to $M_{(i\times j)}$, where $i \times j$ is the total number of items in the dataset. The data processor then dispatches Ms to the third party and meanwhile requests them to proceed with the remaining calculations following the calculation rule A. Upon receiving Ms and A, the third party can compute the entropy value e for each feature based on Eq. 9. Finally, the third party computes the weight W for each feature using Eq. 10, and then transmits the results back to the data processor.

4.2 Naive Bayes Classification

Naive Bayes classification serves as a data mining algorithm that is often used for binary classification tasks under supervised learning [37]. The key point of using Naive Bayes classification is to apply attribute conditional independence assumption [52]. If a dataset has d_f features, the conditional independence assumption can be described as

$$p(x_i|y) = \prod_{j=1}^{d_f} p(x_{ij}|y), \tag{11}$$

where i represents the i^{th} sample and j represents the j^{th} feature in the dataset. In this paper, we mainly focus on the application of Naive Bayes classification on continuous values, a common assumption of which is the continuous values conditioned on the label follow a univariate Gaussian distribution [37]. For simplicity of representation, we assume that the data x has only one dimension, and let μ_k denote the mean and σ_k^2 denote the Bessel corrected variance of the values in x associated with class C_k, respectively. That is,

$$\mu_k = \frac{1}{n_k} \sum_{x_i \in C_k} x_i, \tag{12}$$

where n_k represents the number of samples of class C_k, and

$$\sigma_k = [\frac{1}{n_k - 1} \sum_{x_i \in C_k} (x_i - \mu_k)^2]^{\frac{1}{2}}. \tag{13}$$

Suppose there is a new sample v, so the probability density of v given a class C_k can be computed as follows.

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}. \tag{14}$$

Finally, the category *pred* with the highest probability of occurrence of new sample v is selected, i.e.,

$$pred = \arg\max_{C_k}(p(x = v|C_k)) \tag{15}$$

Naive Bayes classification with ALGOSPEC. The data processor computes μ_k of each feature first using Eq. 12. Then, according to Eq. 13, $(x_i - \mu_k)^2$ is a part of the equation that is viable to apply polynomial approximation. The data processor can have $M_{(i\times j)}$ with the same number as the items in training data and then multiple Ms will be sent to the third party.

When receiving the Ms, the third party computes σ_k for each feature. Then, they calculate the probability of occurrence of each sample under each given category C_k. The third party calculates the probability of occurrence of each sample under different categories by using Eq. 11 and 14, and then sends all computation results to data processor, because data processor needs to estimate the accuracy of the classification.

When the data processor receives the results of $p(x = x_i|C_k)$, they can find the category with the maximum probability of occurrence for each sample, and compare it with its true label. If the accuracy and F1 are satisfactory, the third party can proceed to perform the prediction for new data samples.

Assume the new data sample is v. Similarly, the data processor constructs Ms to represent $(v - \mu_k)^2$ of each feature and sends them to the third party again. Once the third party receives the Ms, $p(x = v|C_k)$ can be calculated by σ_k and sent back. Finally, the data processor finds the category with the highest probability of occurrence of new sample v using Eq. 15.

5 Security Analysis

To establish the security (***Goal #2***) of ALGOSPEC, we leverage a combination of code-based games [20] and decisional Diffie-Hellman (DDH) problems [22]. In this section, we demonstrate how ALGOSPEC maintains the security of the original data against various potential attack scenarios.

5.1 Definitions

ALGOSPEC's selection of n depends on the decisional Diffie-Hellman (DDH) problem, which is a computational problem that arises in the field of cryptography and is often used to assess the security of protocols and systems [22].

Definition 2. *(Decisional Diffie-Hellman problem). The advantage of an adversary* $\mathbf{Adv}^{ddh}_{\mathcal{G},k}(\mathcal{A})$ *in solving the DDH problem relative to* $\mathcal{G}$ *can be defined as the difference between the probability that* $\mathbf{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A})$ *outputs* `true` *and the probability that* $\mathbf{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A})$ *outputs* `true`[3]*, i.e.,.*

$$\begin{aligned}\mathbf{Adv}^{ddh}_{\mathcal{G},k}(\mathcal{A}) = |&\mathcal{P}(\mathbf{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = \texttt{true}) \\ &- \mathcal{P}(\mathbf{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = \texttt{true})|.\end{aligned} \tag{16}$$

In the DDH problem, a finite cyclic group $G = <\mathcal{G}>$ is generated by a generator $\mathcal{G}$, and a random generator $g \xleftarrow{R} \mathbb{G}$ as well as random integers $a, b \xleftarrow{R} \mathbb{Z}^*_p$ are selected uniformly. The goal for the adversary is to distinguish (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, g^c), where c is chosen at random from $\mathbb{Z}^*_p$ [22, 35]. Concretely, adversary will receive g, X, Y and Z, where $X = g^a$, $Y = g^b$ and $Z = g^{ab}$ or $Z = g^c$. If the adversary believes $Z = g^{ab}$, `true` is returned, and `false` otherwise.

The discrete logarithm problem (DLP) is a fundamental mathematical problem with significant applications in scheme security, number theory, and cryptography [40]. ALGOSPEC leverages the inherent computational complexity of the DLP within polynomial time to safeguard the original data during the obfuscation process, ensuring its unrecoverability.

Definition 3. *(Discrete logarithm problem). The DLP is also called Diffie-Hellman key exchange [24]. It is usually defined in the context of finite cyclic groups. In the group* $<\mathcal{G}>$*, given* $x = g^a$ *and* $y = g^b$*, the DLP is to calculate* $g^{ab \bmod p}$ *in polynomial time (PT) [46]. Problems that can be solved in polynomial time are generally seen as tractable, while those that require exponential time or worse are often considered intractable and computationally challenging [19].*

The complexity of solving the DLP relies on the chosen group and its parameters. When the group parameters are appropriately chosen, solving DLP becomes computationally infeasible in PT for prime numbers p. This hardness property is the basis for the security of many security systems.

Definition 4. *(Chosen-plaintext attack). A chosen-plaintext attack (CPA) is a type of cryptographic attack where an adversary has the capability to choose and process specific plaintexts of their choice and then observe the corresponding processed text [9].*

The goal of a CPA is to gain insights into the scheme's security, particularly its vulnerability to various attacks. ALGOSPEC assumes a CPA, which represents a powerful threat model where the adversary has significant control over the inputs, ensuring that the adversary gains no meaningful information about the plaintext from the obfuscated data.

Definition 5. *(Indistinguishability). For an encryption scheme to be considered indistinguishable, for any two plaintexts* m_0 *and* m_1 *chosen by the attacker,* $\mathcal{P}(Enc(m_0) = c) - \mathcal{P}(Enc(m_1) = c) \leq \varepsilon$ *holds, where* m_0 *and* m_1 *are two different plaintexts,* c *is the ciphertext produced by encrypting either one of them and* ε *is a negligible function.*

The most common type of indistinguishability is the *indistinguishability under chosen-plaintext attack* (IND-CPA),

[3] $\mathbf{Exp}^{ddh}_{\mathcal{G},k}(\mathcal{A})$ is the name of a code-based game, and "*ddh*-0" means $c = ab$ in the game, while "*ddh*-1" means $c \xleftarrow{R} \mathbb{Z}^*_p$ in the game.

which is widely used to evaluate the security of cryptographic schemes [25]. IND-CPA assesses the ability of a scheme to protect the confidentiality of plaintext messages in the presence of an active adversary [21]. It provides a strong security guarantee for security schemes and is a critical consideration in the design and evaluation of secure systems.

5.2 *Goal #2* of AlgoSpec

In Section 3.1, we discuss how AlgoSpec can achieve algorithm specificity (***Goal #1***). In this section, we establish that the original data is unrecoverable, in alignment with ***Goal #2***. Two strategies can be used for the attacker to recover the original data. One is to solve an n^{th}-order polynomial based on its degree n, and the other is to analyze the elements in obscured list M and try to recover the original data. We show that when data is obfuscated by AlgoSpec, the attacker cannot effectively recover the original data either way. We list the Theorems to achieve ***Goal #2*** as follows.

Theorem 1. *(Confidentiality of n). Under a CPA, n is not recoverable either from M or the initialization step.*

The proof of Theorem 1 lies in proving that n is confidential at initialization and unrecoverable from M. The former is ensured with generating n based on DDH (Definition 2), and for the latter, AlgoSpec uses a large number (i.e., N) of m_i to hide n. We present the detailed proof based on game playing in Appendix A.2.

Theorem 2. *(Indistinguishability of M). Given M_1 and M_2 that are generated from different degrees n_1 and n_2, respectively, M_1 and M_2 are indistinguishable when given a large N ($N \gg n_1, n_2$).*

Theorem 1 demonstrates that M reveals no information for the adversary to infer n. We utilize this to establish the validity of Theorem 2. Our proof is based on two sub-games of generating M, one using the real n and the other using another degree that is randomly generated. We prove that the difference of the probabilities that the attacker successfully differentiates M in this two sub-games is a negligible function. The detailed proof is listed in Appendix A.3.

Main Theorem. *(Unrecoverability of ℓ). Under a CPA, the original data ℓ is not recoverable by analyzing M or solving the polynomial by obtaining the degree n.*

The main theorem is also proved using code-based game playing. We define a challenge representing the obscured list M that is known by the attacker, and construct a game which asks the attacker to guess whether this challenge represents M or a randomly generated list. The constructed game comprises four sub-games under IND-CPA, and each pair of adjacent sub-games are initialized with difference in only one setting. We generate the challenge C in the first sub-game using the real ℓ, and generate another challenge C' in the last sub-game using a randomly generated ℓ'. By demonstrating that consecutive sub-games satisfy $|\mathcal{P}(G_i(\mathcal{A})=1) - \mathcal{P}(G_{i+1}(\mathcal{A})=1)| \leq \varepsilon$, ($i \in [0,2]$), the first and last sub-games also satisfy $|\mathcal{P}(G_0(\mathcal{A})=1) - \mathcal{P}(G_3(\mathcal{A})=1)| \leq \varepsilon$. The detailed proof is presented in Appendix A.4.

Table 3: Datasets used in this paper.

Name	Size	Precision	Apply	Expanded
WH	149	3-digit	EM	
WH ep1	10^3	3-digit	EM	✓
WH ep2	10^4	3-digit	EM	✓
WH ep3	10^5	3-digit	EM	✓
WH ep4	10^6	3-digit	EM	✓
breast cancer	569	6-digit	NB	
heart disease	1,025	single-prec.	NB	
hypertension	26,058	single-prec.	NB	
hypertension ep	104,232	single-prec.	NB	✓

WH: 2021 world happiness report

5.3 Discussion on n's Confidentiality

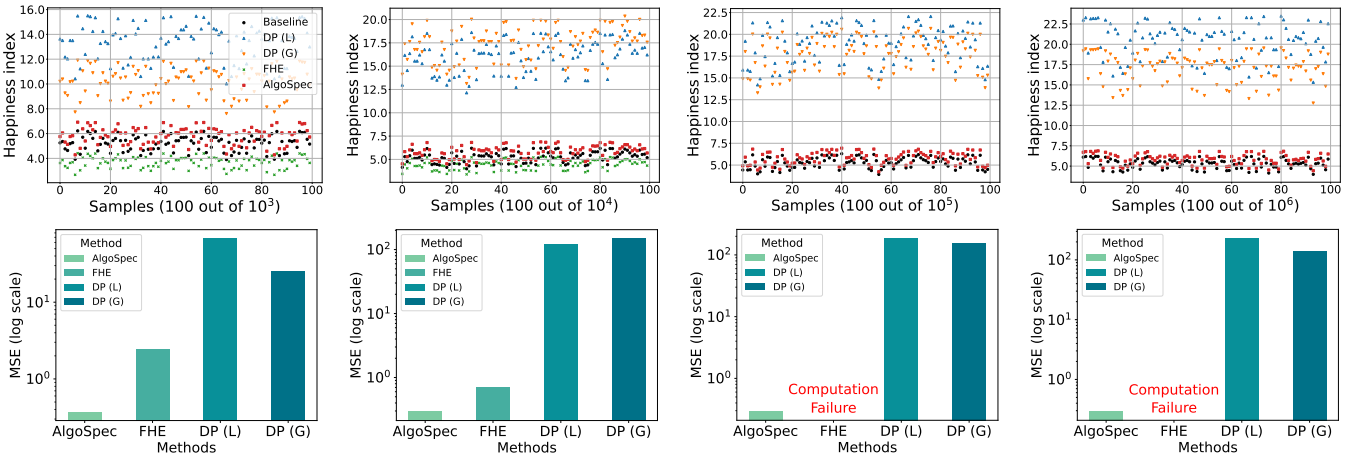
We have proved that ℓ is unrecoverable when n is sufficiently large through code-based game-playing. However, even when n is not exceptionally large, the protection for ℓ can still be ensured. The only way for the adversary to get ℓ is to solve the high-order polynomial as discussed in Section 3.1. When the highest order of the polynomial is greater than 4, solving such a high-order polynomial is an NP-hard problem [44]. Importantly, the adversary must solve high-order polynomials for all ns they selected, which means they face a plethora of NP-hard problems. Therefore, a relatively small n is sufficient in practical use. We empirically find that an n greater than 500 can lead to a precise approximation of complicated algorithms.

6 Evaluation

To evaluate the performance of AlgoSpec, we apply it to the entropy method and Naive Bayes classification. We conduct benchmarking and scalability analysis, to provide a comprehensive assessment of its capabilities.

6.1 Experimental Settings

Datasets. We mainly use the breast cancer [1], heart disease [2], 2021 world happiness report [5], and hypertension [10] datasets to assess the efficacy of AlgoSpec. To provide a thorough evaluation of our methodology across varied dataset sizes, we expand our initial dataset. This allows us to analyze data magnitudes spanning from 10^2 to 10^6. The details of the datasets used in our evaluation is presented in Table 3.



(a) scale of 10^3 dataset (b) scale of 10^4 dataset (c) scale of 10^5 dataset (d) scale of 10^6 dataset

Figure 4: Accuracy and MSE of each method on entropy method through different scales of datasets from 10^3 to 10^6.

Running environment. Our evaluation is conducted on a workstation of Ubuntu 22.04.3 LTS x86_64 as the operating system, and AMD Ryzen Threadripper PRO 5965WX with x86_64 architecture, 48 cores, and 3.800GHz CPU.

Settings. To evaluate the performance of ALGOSPEC side-by-side, we compare it to the baseline, FHE, and DP, where their settings are presented as follows.

Baseline. The baseline is calculated using the authorized algorithm directly, without encryption or preprocessing.

FHE. FHE is a widely-used privacy protection method and has a similar threat model to ALGOSPEC. Despite its limitations in fully addressing the purpose limitation issue, it holds relevance in ***Scenario #2*** (Figure 1.(b)), where the third party can apply any algorithm on the cipher, but has no decryption key to see the computation results. Therefore, FHE is applied to compare with ALGOSPEC.

We choose the currently dominant Python homomorphic encryption library, pi-HEaaN [7], as the main framework, and use Taylor expansion for the non-linear part of an algorithm. To eliminate the computational noise through FHE, we set the scale factor as $\frac{1}{9}$ due to its best performance.

DP. DP is another widely-used privacy-preserving method. It is known for the efficiency due to the lightweight additive noise strategy. Therefore, we select it to benchmark the efficiency of ALGOSPEC. When using DP, we employ two noise mechanisms, i.e., the Laplace mechanism and the Gaussian mechanism. For Laplace mechanism, we set both ε_{dp} and sensitivity to 1.0; for Gaussian mechanism, we set ε_{dp}, δ_{dp}, and sensitivity all to 1.0.

Secure function evaluation (SFE). SFE [33] allows two parties to jointly conduct computation. Thus, we also explore applying SFE for algorithm specificity. We apply two widely-used libraries, namely the garbled-circuit [4] and TenSEAL [3], to the entropy method and the Naive Bayes classification. However, the former fails to split the raw data into two Yao's circuits, as it necessitates the design of intricate circuits to represent logarithmic operations as well as probabilistic calculations. The latter runs into the out-of-memory exception due to the computational complexity. Thus, the results on SFE are not reported in this paper.

6.2 Benchmarking

We benchmark the performance of ALGOSPEC on entropy method and Naive Bayes classification against baseline, FHE, DP with Laplace mechanism, and DP with Gaussian mechanism. The scale of the dataset we use is controlled at 10^4 on entropy method and 10^3 on Naive Bayes classification.

Entropy method. To compare ALGOSPEC's accuracy against others on the entropy method, we first compute the output of the 2021 World Happiness Report ep2, producing the Happiness Index as the accuracy indicator. The results are illustrated in Figure 4.(b). Notably, ALGOSPEC (■) significantly outperforms DP with Laplace or Gaussian mechanism (▲, ▼), and is closer to the baseline (●) than FHE (×), as reflected by the distances from the baseline. To quantify the distance, we utilize the Mean Squared Error (MSE) as an evaluation metric, as illustrated in the four figures at the bottom of Figure 4. Overall, ALGOSPEC excels on the entropy method in terms of accuracy. Unsurprisingly, both DP mechanisms exhibit relatively high MSE, exceeding 100, as the noise added undermines the accuracy of the algorithm. The MSE of FHE is also much higher than ALGOSPEC, because the intermediate noise produced by FHE during computation can significantly affect the classification task, specifically the process of operating the `argmax` function [36].

Next, we evaluate the efficiency of ALGOSPEC. For all methods, we gauge the total computation time, which encompasses the duration from data retrieval to the completion of

Table 4: Efficiency of different scale datasets from 10^3 to 10^6 on entropy method.

	10^4			10^3			10^5			10^6		
	PT	ET	TT	PT	ET	TT	PT	ET	TT	PT	ET	TT
Baseline	0.000	0.235	0.235	0.000	0.028	0.028	0.000	2.168	2.168	0.000	20.793	20.793
DP (L)	1.007	0.218	1.225	0.097	0.025	0.122	10.194	2.136	12.330	101.904	21.684	123.588
DP (G)	1.030	0.219	1.249	0.128	0.023	0.151	10.137	2.141	12.278	104.117	21.016	125.133
FHE	0.134	487.663	487.797	0.015	5.717	5.732	-	-	-	-	-	-
ALGOSPEC	36.378	1.069	37.447	3.601	0.109	3.710	365.054	10.409	376.463	3675.673	102.339	3778.012

PT: Processing Time **ET**: Execution Time **TT**: Total Time

Table 5: Test accuracy and test F1 of different scale datasets from 10^2 to 10^5 on Naive Bayes classification.

	Heart Disease (10^3)		Breast Cancer (10^2)		Hypertension (10^4)		Hypertension ep (10^5)	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
Baseline	0.844	0.851	0.921	0.886	0.834	0.854	0.832	0.850
DP (L)	0.698	0.705	0.904	0.853	0.747	0.780	0.751	0.782
DP (G)	0.790	0.786	0.895	0.838	0.791	0.818	0.799	0.824
FHE	0.473	0.460	0.465	0.371	0.543	0.594	0.513	0.540
ALGOSPEC	0.824	0.833	0.921	0.877	0.812	0.829	0.818	0.832

result computation. Given the involvement of a data processor and a third party in our setup, we further break down the total time into processing time and execution time. Processing time signifies the time the data processor takes on data manipulation. DP's processing time includes the duration of adding noise. FHE's processing is the data encryption and key generation. ALGOSPEC 's processing encompasses polynomial approximation and obscured list construction. On the other hand, execution time corresponds to the duration the third party dedicates to carrying out calculations for the final results. The first segment (benchmark) of Table 4 offers insights into the time taken by different methods during the execution of the entropy method, enabling a comprehensive assessment of efficiency. The baseline method requires no data processing, and hence, there is no processing time associated with it. Regarding the processing time, ALGOSPEC requires slightly more time due to its preprocessing of both the original data and the intended algorithm. Concerning execution time, our approach falls between DP and FHE but is significantly faster than FHE, completing the task about 487 times shorter. Similarly for the total time, our method also significantly outperforms FHE (nearly 13 times faster).

Naive Bayes classification. We use the heart disease dataset to evaluate the performance of ALGOSPEC on Naive Bayes classification. We take test accuracy and test F1 scores as performance indicators for the evaluation. The first segment (benchmark) of Table 5 illustrates the performance of each method. By using the baseline as the reference, and comparing the accuracy and F1 of DP, FHE, and our method, we observe that ALGOSPEC exhibits the closest accuracy and F1 to the baseline, with both metrics decreasing by less than 0.02. DP, regardless of the noise mechanism used, lags behind ALGOSPEC in terms of both accuracy and F1. FHE performs the poorest, displaying the lowest test accuracy and F1. To illustrate the performance comparison, we use the receiver operating characteristic (ROC) curves and calculate the area under the ROC curve (AUC). Figure 5.(a) presents these curves and AUC values for each method. The red line represents the ROC curve for ALGOSPEC, which boasts the closest AUC to the baseline at 0.8272. It is evident that the AUC of our method encompasses those of the other methods, establishing our method as the top performer on accuracy.

Moving on to the efficiency evaluation, we continue to divide the total time into processing time and execution time. Regarding efficiency comparison, which is illustrated in the lead-in segment (benchmark) of Table 6, our results align with those from the entropy method. Although ALGOSPEC requires slightly longer processing time, its execution and total time are significantly superior, nearly 7 times and 4 times faster than FHE, respectively.

ALGOSPEC demonstrates superior performance in terms of accuracy, as evaluated on both the entropy method and Naive Bayes classification, surpassing other mainstream privacy-preserving approaches. For efficiency, ALGOSPEC is between FHE and DP, but given that neither DP nor FHE can maintain a consistent accuracy rate, we believe that it is worthwhile to sacrifice a certain amount of efficiency for a stable and high accuracy of our method.

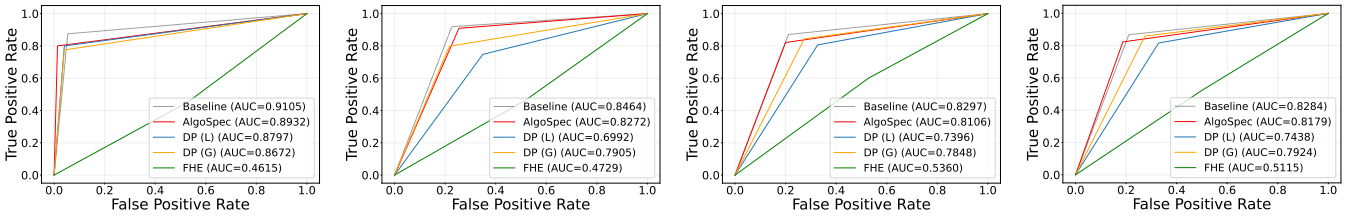
6.3 Scalability

To explore whether ALGOSPEC performs effectively across datasets of various scales, we assess its scalability by utilizing datasets ranging from 10^2 to 10^6.

Table 6: Efficiency of different scale datasets from 10^2 to 10^5 on Naive Bayes classification.

	Heart Disease (10^3)			Breast Cancer (10^2)			Hypertension (10^4)			Hypertension ep (10^5)		
	PT	ET	TT	PT	ET	TT	PT	ET	TT	PT	ET	TT
Baseline	0.000	0.513	0.513	0.000	0.467	0.467	0.000	2.735	2.735	0.000	9.088	9.088
DP (L)	0.082	0.575	0.657	0.104	0.544	0.648	2.034	2.362	4.396	7.750	8.016	15.766
DP (G)	0.059	0.576	0.635	0.077	0.545	0.622	1.421	2.316	3.737	5.448	7.446	12.894
FHE	0.040	9.980	10.020	0.269	3.097	3.366	0.243	196.428	196.672	0.909	788.161	789.070
AlgoSpec	1.432	1.520	2.952	2.011	1.645	3.656	35.915	41.415	77.330	143.045	175.288	318.333

PT: Processing Time **ET**: Execution Time **TT**: Total Time



(a) Heart disease (10^3) (b) Breast cancer (10^2) (c) Hypertension (10^4) (d) Hypertension ep (10^5)

Figure 5: ROC curve and AUC of each method on Naive Bayes classification through different scales of datasets from 10^2 to 10^5.

Entropy method. We use datasets ranging from 10^3 to 10^6 for the entropy method. Figure 4 displays the computation of the happiness index by each method and the Mean Squared Error (MSE) between each method's results and the baseline for datasets of different sizes. Table 4 provides an overview of the efficiency of each method on datasets of varying scales. Notably, as the dataset size exceeds 10^4, FHE encounters computational challenges due to the complexity of its intermediate generated noise. In contrast, AlgoSpec exhibits exceptional scalability, making it capable of handling large-scale datasets.

Naive Bayes classification. We choose the scale of datasets from 10^2 to 10^5 on Naive Bayes classification. Table 5 shows the test accuracy and test F1, while Figure 5 illustrates the ROC curve and AUC of each method on different scales of datasets. We also compare each method's time efficiency across various dataset scales, as shown in Table 6.

AlgoSpec is highly scalable since it can operate on datasets of different sizes ranging from 10^2 to 10^6, which is ideal for our settings of the threat model.

7 Discussions

In this section, we present a discussion on AlgoSpec's generalization for simple and complex algorithms, as well as the challenges of deploying AlgoSpec in practice.

Generalization. Given the target application scenarios of AlgoSpec, the algorithms are typically not simplistic. AlgoSpec could also apply an alleviation for simple algorithms, by covering the simple algorithm with random calculation at an invalid input range. Taking $y = x^2$ as an example, after determining the range of the input (Section 3.1), it keeps the segment of $y = x^2$ within this range. For the remaining, it generates a random complex calculation and combines it with the kept segment. This solution allows AlgoSpec to keep a high degree of n for simple algorithms. For complex algorithms such as neural networks, AlgoSpec faces challenges during the iterative backpropagation. It requires model parameters from previous iterations to approximate new polynomials for the current iteration, meaning that its approximation has to be parameterized. We take the extension of AlgoSpec for parameterized algorithms as future work.

Deployment. When deploying AlgoSpec in real-world scenarios, it entails the client (in ***Scenario #1***) and the data processor (in ***Scenario #2***) to own a certain level of computational capabilities for the polynomial approximation and the obscured list construction. The obscured list generated by AlgoSpec enlarges the original data into an N dimension vector, causing increased storage and network communication costs from $O(1)$ to $O(N)$.

8 Related Work

There are several data protection methods [30, 43, 45, 53, 54] that data processors could use to protect the original data and attempt to meet the purpose limitation principle when sending the data to third parties. We categorize the current mainstream data protection methods into three groups, including *cryptography-based*, *noise-based*, and *algorithm-based*. In this section, we summarize each of them.

Cryptography-based data protection. The cryptography-based data protection that can be used for purpose limitation is fully homomorphic encryption (FHE). Popescu et al. [45]

propose a way of encrypting rational numbers using FHE by decomposing the rational numbers to various polynomials with the order from "0" to "$-n$", which protects rational number data in the medical field. Mukherjee et al. [43] develop a multi-key FHE scheme, which is a framework that allows different participants to use different encryption keys following a fixed computation method, to finally generate the results. Zhao et al. [54] give a solution to the big data privacy problem by deploying FHE to cloud computing, which enables third parties to execute relatively large-scale operations. Although FHE has strong data protection capabilities, it cannot achieve algorithm specificity, particularly, in ***Scenario #1***. Furthermore, its computational complexity is extremely high, requiring a large number of computational units to perform otherwise simple operations [16].

Noise-based data protection. Adding noise or perturbations to the original data is another approach to obfuscate data. One of the widely used noise-based methods is differential privacy (DP), which adds noise via the Laplace mechanism or Gaussian mechanism [30]. DP controls the intensity of the noise by setting different privacy loss parameters, i.e., ε_{dp} and δ_{dp}, as well as sensitivity. In general, the larger the values of ε_{dp} and δ_{dp}, the less noise is added to the data and the weaker the protection. In order to achieve satisfactory computational results with the protection of data security, the values of sensitivity, ε_{dp}, and δ_{dp} are set to 1.0. Cheu et al. [26] propose a scalable and robust protocol with DP by randomizing a set of user-supplied information through anonymous channels, to hide the sensitive information of private data. Wang et al. [50] present a DP method that improves the application of using local differential privacy to multidimensional data collection, so that complex computation can be performed on high-dimensional data. DP is easy to apply, but it may lead to sharp accuracy loss to the intended algorithm.

Algorithm Specificity. Limited research has been dedicated to purpose limitation, resulting in a limited number of methods capable of achieving algorithm specificity. Two recent studies might be adapted to the purpose limitation principle these days. Zhang et al. [53] add targeted noise to images to make them work only for specific tasks, thus protecting them from misuse and unauthorized online searches. Syed-Winkler et al. [47] present a system model for enforcing purpose limitation based on data tagging and attribute-based encryption, ensuring access control based on the purpose of a service. However, although these existing data protection methods can protect the data owners' original data to a certain extent, none of them can really satisfy algorithm specificity under the purpose limitation principle. Thus, to the best of our knowledge, we are the first to discover how to achieve algorithm specificity and provide a novel solution, AlgoSpec, through the lens of algorithm level.

9 Conclusion

We propose AlgoSpec, a solution of algorithm specificity that implements the purpose limitation principle. AlgoSpec's approach is based on the properties of polynomial approximation and complex mathematical problems, so as to convert the intended algorithm and the original data into an obfuscated format. It ensures two crucial goals for the purpose limitation principle. The original data are not recoverable by polynomial time attacks, and the obfuscated data can only be computed with authorized algorithms to generate desired results. We provide a formal proof on the security of AlgoSpec, and experimentally evaluate its accuracy and efficiency performance on the entropy method and Naive Bayes Classification. Our studies demonstrate that AlgoSpec balances computational efficiency with a high level of accuracy. Additionally, its high scalability also enables its applicability in a great range of scenarios and applications. We hope our study can attract more attention to the purpose limitation principle, going beyond the transparency principle that has become common practice nowadays.

Acknowledgement

We thank the anonymous reviewers and shepherd for their insightful comments to improve this manuscript. This work is partially supported by Australian Research Council Discovery Projects (DP230101196, DP240103068) and Ant Group. Minhui Xue is supported by CSIRO – National Science Foundation (US) AI Research Collaboration Program.

References

[1] Breast cancer dataset. https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset, 2015.

[2] Heart disease dataset. https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset, 2019.

[3] A library for homomorphic encryption operations on tensors. https://pypi.org/project/tenseal/0.1.0a0/, 2020.

[4] Secure multi-party computation. https://github.com/ojroques/garbled-circuit, 2020.

[5] World happiness report 2021. https://www.kaggle.com/datasets/ajaypalsinghlo/world-happiness-report-2021, 2021.

[6] Amazon.com.au privacy notice. https://www.amazon.com.au/gp/help/customer/display.html?nodeId=GX7NJQ4ZB8MHFRNJ, 2022.

[7] pi-heaan 3.4.1.3. https://pypi.org/project/pi-heaan/, 2022.

[8] California consumer privacy act (ccpa). https://oag.ca.gov/privacy/ccpa, 2023.

[9] Chosen plaintext attack. https://nordvpn.com/cybersecurity/glossary/chosen-plaintext-attack, 2023.

[10] Diabetes, hypertension and stroke prediction. https://www.kaggle.com/datasets/prosperchuks/health-dataset?select=hypertension_data.csv, 2023.

[11] General data protection regulation (gdpr) - official legal text. https://gdpr-info.eu, 2023.

[12] General data protection regulation (gdpr) - principles relating to processing of personal data. https://gdpr-info.eu/art-5-gdpr/, 2023.

[13] Principle (b): Purpose limitation - what is the purpose limitation principle? https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources, 2023.

[14] The Privacy Act. https://www.oaic.gov.au/privacy/privacy-legislation/the-privacy-act, 2023.

[15] What is the general data protection regulation (gdpr)? https://en-gb.facebook.com/business/gdpr, 2023.

[16] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: theory and implementation. *ACM Computing Survey*, 51(4):1–35, 2018.

[17] Dorit Aharonov, Xun Gao, Zeph Landau, Yunchao Liu, and Umesh Vazirani. A polynomial-time classical algorithm for noisy random circuit sampling. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 945–957, 2023.

[18] Mehrdad Aliasgari, Marina Blanton, Yihua Zhang, and Aaron Steele. Secure computation on floating point numbers. *Cryptology ePrint Archive*, 2012.

[19] Marco Ancona, Cengiz Oztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *International Conference on Machine Learning (ICML)*, pages 272–281, 2019.

[20] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, pages 409–426, 2006.

[21] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1175–1191, 2017.

[22] Dan Boneh. The decision diffie-hellman problem. In *International Algorithmic Number Theory Symposium (ANTS)*, pages 48–63, 1998.

[23] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[24] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably secure authenticated group diffie-hellman key exchange. *ACM Transactions on Information and System Security*, 10(3), 2007.

[25] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. Relationships between quantum ind-cpa notions. In *Theory of Cryptography Conference (TCC)*, pages 240–272, 2021.

[26] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, pages 375–403, 2019.

[27] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. Polynomial time attack on wild mceliece over quadratic extensions. *IEEE Transactions on Information Theory*, 63(1):404–427, 2016.

[28] Bert Den Boer. Diffie-hellman is as strong as discrete log for certain primes. In *Annual International Conference on the Theory and Application of Cryptography (Eurocrypt)*, pages 530–539, 1988.

[29] Nico Döttling, Jörn Müller-Quade, and Anderson CA Nascimento. Ind-cca secure cryptography based on a variant of the lpn problem. In *18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 485–503, 2012.

[30] Cynthia Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1–12, 2006.

[31] Daniele Funaro. *Polynomial approximation of differential equations*, volume 8. Springer Science & Business Media, 2008.

[32] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM symposium on Theory of Computing (STOC)*, pages 169–178, 2009.

[33] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *in 6th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 162–177, 2000.

[34] Bin Jia, Xiaosong Zhang, Jiewen Liu, Yang Zhang, Ke Huang, and Yongquan Liang. Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot. *IEEE Transactions on Industrial Informatics*, 18(6):4049–4058, 2021.

[35] Antoine Joux and Kim Nguyen. Separating decision diffie–hellman from computational diffie–hellman in cryptographic groups. *Journal of cryptology*, 16:239–247, 2003.

[36] Nikola Jovanovic, Marc Fischer, Samuel Steffen, and Martin Vechev. Private and reliable neural network inference. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1663–1677, 2022.

[37] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[38] Hans Lausch and Wilfred Nobauer. *Algebra of polynomials*. Elsevier, 2000.

[39] Ueli M Maurer and Stefan Wolf. The relationship between breaking the diffie–hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999.

[40] Kevin S McCurley. The discrete logarithm problem. In *Proceedings of Symposia in Applied Mathematics*, volume 42, pages 49–74, 1990.

[41] Hrushikesh Narhar Mhaskar. *Introduction to the theory of weighted polynomial approximation*, volume 7. World Scientific, 1996.

[42] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference (TCC)*, pages 133–151, 2004.

[43] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt)*, pages 735–763, 2016.

[44] V Th Paschos. An overview on polynomial approximation of np-hard problems. *Yugoslav Journal of Operations Research*, 19(1), 2016.

[45] Andreea Bianca Popescu, Ioana Antonia Taca, Cosmin Ioan Nita, Anamaria Vizitiu, Robert Demeter, Constantin Suciu, and Lucian Mihai Itu. Privacy preserving classification of eeg data using machine learning and homomorphic encryption. *Applied Sciences*, 11(16):7360, 2021.

[46] Nigel Paul Smart et al. *Cryptography: an introduction*, volume 3. McGraw-Hill New York, 2003.

[47] Sarah Syed-Winkler, Sebastian Pape, and Ahmad Sabouri. A data protection-oriented system model enforcing purpose limitation for connected mobility. In *Proceedings of the 6th ACM Computer Science in Cars Symposium (CSCS)*, pages 1–11, 2022.

[48] Yuerong Tong, Lina Yu, Sheng Li, Jingyi Liu, Hong Qin, and Weijun Li. Polynomial fitting algorithm based on neural network. *ASP Transactions on Pattern Recognition and Intelligent Systems*, 1(1):32–39, 2021.

[49] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Sok: Differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy (S&P)*, pages 354–371, 2020.

[50] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649, 2019.

[51] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Computing survey*, 53(4):1–35, 2020.

[52] Harry Zhang. The optimality of naive bayes. *in Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 2004.

[53] Peng-Fei Zhang, Guangdong Bai, Hongzhi Yin, and Zi Huang. Proactive privacy-preserving learning for cross-modal retrieval. *ACM Transactions on Information Systems*, 41(2):1–23, 2023.

[54] Feng Zhao, Chao Li, and Chun Feng Liu. A cloud computing security solution based on fully homomorphic encryption. In *16th International Conference on Advanced Communication Technology (ICACT)*, pages 485–488, 2014.

Algorithm 1: Workflow of ALGOSPEC

Input: Original data $\mathbb{D}_f^{(s)}$, $f \in [0,k]$, $s \in [0,l]$, $k \geq 0$, $l \geq 0$, $k,l \in \mathbb{R}$, Original algorithm Fx
Output: Final result r

Initialize: λ, d, γ,
procedure *construct*($\alpha s, \beta s, \lambda, d, N, \mathbb{D}_f^{(s)}$)
  for all $\mathbb{D}_i^{(j)}$ **in** $\mathbb{D}_f^{(s)}$ **do**
    $M' \leftarrow construct\_noncons(\alpha s, \lambda_n, d_n, \mathbb{D}_i^{(j)})$
    $B \leftarrow construct\_cons(\beta s, pairs, times)$
    $M_i^{(j)} \leftarrow combine(M', B)$
  end for
  return $\{M_0^{(0)}, \ldots, M_i^{(j)}, \ldots, M_f^{(s)}\}$
end procedure
for $i := 0$ **to** k **do**
  $coefs[i] \leftarrow poly\_approx(Fx, ns[i])$
end for
$\alpha s \leftarrow sep\_coef(coefs)$
$\beta s \leftarrow sep\_cons(coefs)$
$Ms \leftarrow construct(\alpha s, \beta s, \lambda, d, N, \mathbb{D}_f^{(s)})$
$r \leftarrow compute(Ms, A)$
return r

Table 7: Code-based game-playing $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh}(\mathcal{A})$ to prove n is indistinguishable

Game $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$:	**Game** $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A})$:
Proc Initialize (k)	Proc Initialize (k)
$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$	$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$
$g \xleftarrow{R} G^*$	$g \xleftarrow{R} G^*$
$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \xleftarrow{R} g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \xleftarrow{R} g^x$
$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \xleftarrow{R} g^y$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \xleftarrow{R} g^y$
$z \xleftarrow{R} xy \bmod p$; $Z \xleftarrow{R} g^z$	$z \xleftarrow{R} \mathbb{Z}_p^*$; $Z \xleftarrow{R} g^z$
Return (G,g,X,Y,Z)	*Return* (G,g,X,Y,Z)
Proc Finalize (β')	Proc Finalize (β')
Guess:	Guess:
if $\begin{cases} Z = g^{\log_g X+Y}, & set\ \beta' = 1 \\ Z \neq g^{\log_g X+Y}, & set\ \beta' = 0 \end{cases}$	if $\begin{cases} Z = g^{\log_g X+Y}, & set\ \beta' = 1 \\ Z \neq g^{\log_g X+Y}, & set\ \beta' = 0 \end{cases}$
Return (β')	*Return* (β')

A Appendix

A.1 Main Algorithm of ALGOSPEC

We represent the workflow of ALGOSPEC in Algorithm 1.

A.2 Proof of Theorem 1

We prove Theorem 1 by proving the correctness of Lemma 1 and Lemma 2.

Lemma 1. *(n is initialized indistinguishable). Let* $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh}(\mathcal{A})$ *be a code-based game-playing.* $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$ *and* $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A})$ *are the sub-games in* $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh}(\mathcal{A})$*, which is illustrated in Table 7. In Proc Initialize (k) of both sub-games, the only difference is* $z \xleftarrow{R} xy \bmod p$ *in* $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$ *and* $z \xleftarrow{R} \mathbb{Z}_p^*$ *in* $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A})$*. In proc Finalize* (β')*, the adversary guesses if* $Z = g^{\log_g X+Y}$ *and returns* $\beta' = 1$ *if the answer is yes, otherwise* $\beta' = 0$ *in the two sub-games. We have that n is indistinguishable if*

$$\begin{aligned} \mathbf{Adv}_{\mathcal{G},k}^{ddh}(\mathcal{A}) = |&\mathcal{P}(\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A}) = 1) \\ &- \mathcal{P}(\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A}) = 1)| \leq \varepsilon, \end{aligned} \tag{17}$$

where ε is negligible.

Proof of Lemma 1. The degree n of the polynomial is generated based on DDH (Definition 2), which can be considered as DLP [28, 39] (Definition 3). Let $G = <\mathcal{G}>$ be the finite cyclic group; p is the prime order; $k = |p|$ be the security parameter; $g \xleftarrow{R} G^*$ be the generated element. We generate n by $x \xleftarrow{R} \mathbb{Z}_p^*$, $y \xleftarrow{R} \mathbb{Z}_p^*$, $z \xleftarrow{R} xy \bmod p \Rightarrow n = g^z$.

To prove that n is indistinguishable, we construct a code-based game-playing $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh}(\mathcal{A})$. Table 7 shows the entire game consisting of game $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$ and game $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A})$. Each sub-game contains two processes, the initialization process *Proc Initialize* (k) and finalization process *Proc Finalize* (β'), where β' is a flag that can be set to 1 or 0 based on the adversary guess.

We first design *Proc Initialize* (k) of the two sub-games. It generates a cyclic group G with prime order p using group generator $\mathcal{G}$. Then, it randomly selects a random generator g from G^* and x, y from $\mathbb{Z}_p^*$, and sets $X \xleftarrow{R} g^x$ as well as $Y \xleftarrow{R} g^y$. The only difference between $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$ and $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A})$ is illustrated by the red rectangle. In $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$, z is the real $z \xleftarrow{R} xy \bmod p$ that we use to generate n, while in $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A})$, z is randomly selected from $\mathbb{Z}_p^*$. Then we set $Z \xleftarrow{R} g^z$ and return (G, g, X, Y, Z) in both sub-games. In the *Proc Finalize* (β'), the adversary ends the game by guessing whether $Z = g^{\log_g X+Y}$. If the adversary guesses $Z = g^{\log_g X+Y}$, they set $\beta' = 1$, otherwise they set $\beta' = 0$.

In the *Proc Initialization* (k), there is a DLP. By giving g, X, Y and Z, one cannot distinguish whether $z \xleftarrow{R} xy \bmod p$ or $z \xleftarrow{R} \mathbb{Z}_p^*$. Thus, in both sub-games, based on the returned (G, g, X, Y, Z), the adversary is not able to find the relationship between Z and X, Y, which means when given $Z = g^{\log_g X+Y}$ and $Z \neq g^{\log_g X+Y}$, the adversary can only randomly select one, namely a random guess. As a result, the advantage $\mathbf{Adv}_{\mathcal{G},k}^{ddh}(\mathcal{A})$ of the adversary to win the game $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh}(\mathcal{A})$ is negligible.

Lemma 2. *(n is unrecoverable from M). Let F be a polynomial with the degree n. Let M denote a list containing* $\frac{N}{2}$ *pairs* $(N \gg 2n)$ *with almost evenly distribution, which follows* γ *distances.* γ *is well-designed by* $\gamma = \rho + \omega$*, where* ρ *is the number of distances of n pairs and* ω *is the new distances to be created* $(0 \leq \omega \leq \frac{N}{4} - \rho)$*. Let A denote the calculation rule in* ALGOSPEC*. The CPA is unable to discern n of the polynomial when provided with M.*

Table 8: Code-based game-playing $\boldsymbol{Exp}^{ddh}_{\mathcal{G},k,Gen()}(\mathcal{A})$ to prove M is indistinguishable

Game $\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A})$:	**Game** $\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A})$:
Proc Initialize (k,ℓ,F)	Proc Initialize (k,ℓ,F)
$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$	$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$
$g \xleftarrow{R} G^*$; $N \xleftarrow{R} \mathbb{Z}^{\gg n}$	$g \xleftarrow{R} G^*$; $N \xleftarrow{R} \mathbb{Z}^{\gg n}$
$\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$
$y \xleftarrow{R} \mathbb{Z}_p^*$	$y \xleftarrow{R} \mathbb{Z}_p^*$
$n \xleftarrow{R} g^{xy \bmod p}$	$n \xleftarrow{R} \mathbb{Z}_p^*$
$M \xleftarrow{R} Gen(\ell,n,N,\gamma,F)$	$M \xleftarrow{R} Gen(\ell,n,N,\gamma,F)$
Return M	*Return* M
Proc Finalize (β')	Proc Finalize (β')
Guess:	Guess:
if $\begin{cases} n = g^{xy}, & set\ \beta' = 1 \\ n \neq g^{xy}, & set\ \beta' = 0 \end{cases}$	if $\begin{cases} n = g^{xy}, & set\ \beta' = 1 \\ n \neq g^{xy}, & set\ \beta' = 0 \end{cases}$
Return (β')	*Return* (β')

To facilitate the proof, we present a supporting claim that formulates Lemma 2 in a mathematical way. n in the claim represents the degree of the polynomial; $\frac{N}{2}$ are exactly the number of pairs in M; following the construction of pattern, we set γ (almost evenly distributed) distances in the claim with the same ρ and ω. Subsequently, we establish the correctness of Lemma 2 by demonstrating the validity of Claim 1.

Claim 1. *Given n pairs following ρ distances and $\left(\frac{N}{2}-n\right)$ pairs that are constructed to follow $\gamma=\rho+\omega$ distances, $\frac{N}{2}$ pairs that are evenly distributed can compose a list M, such that despite the attacker possessing M, n remains indistinguishable.*

Proof of Claim 1. Given $\frac{N}{2}$ pairs and γ distances ($n \ll \gamma < \frac{N}{2}$), there must be at least one distance containing multiple pairs, according to the pigeonhole principle. Therefore, the range of distance in n pairs is $\rho \in [1,n]$ Additionally, we introduce λ and d to adjust the number of distances, ensuring an almost uniform distribution within each distance. M comprises $\frac{N}{2}$ pairs distributed across γ patterns. Considering $\rho \in [1,n]$ and $\omega \in [0, \frac{N}{4}-\rho]$, it follows that $\gamma \in [1, n+\frac{N}{4}-\rho]$. In practical scenarios, where $\frac{N}{2} \gg n \geq \rho$, if $\gamma \leq n$ meaning that $\gamma \in [1,n]$, the adversary needs to attempt $(\frac{N}{2}-\gamma+1)$ times to determine n; if $\gamma > n$ meaning that $\gamma \in (n, n+\frac{N}{4}-\rho]$, the adversary needs to attempt $(\gamma-1)$ times. However, due to the even distribution, the adversary remains unaware of whether $\gamma < n$ or $\gamma > n$, necessitating $\frac{N}{2}$ $(\frac{N}{2}-\gamma+1+\gamma-1)$ attempts, which is impossible within PT. Consequently, even if M is disclosed, n remains unrecoverable.

A.3 Proof of Theorem 2

Table 8 shows the code-based game playing $\boldsymbol{Exp}^{ddh}_{\mathcal{G},k,Gen()}(\mathcal{A})$ used in our proof. Its construction follows the intuition that two Ms generated using different n are indistinguishable. $\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A})$ and $\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A})$ are the sub-games. In $\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A})$, n is set by $n \xleftarrow{R} g^{xy \bmod p}$; in $\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A})$, n is set by $n \xleftarrow{R} \mathbb{Z}_p^*$. Both sub-games return M with their corresponding n in *Proc Initialize*(k, ℓ, F). The adversary is not able to distinguish which M is generated from the real n. M is indistinguishable if

$$\begin{aligned}\mathbf{Adv}^{ddh}_{\mathcal{G},k,Gen()}(\mathcal{A}) = |\mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A})=1\right) \\ -\mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A})=1\right)| \leq \varepsilon\end{aligned} \tag{18}$$

The *Proc Initialize*(k,ℓ,F) of the two sub-games is designed to generate M. Their only difference is marked with red rectangles. In $\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A})$, n is the real one generated by $n \xleftarrow{R} g^{xy \bmod p}$, whereas in $\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A})$, n is generated randomly. In the *Proc Finalize*(β), once the adversary has the Ms returned by the two games, they need to determine which M is generated by the real n. In other words, they have to guess whether $n = g^{xy}$[4] or not based on the returned Ms. If they believe $n = g^{xy}$, they set $\beta' = 1$, otherwise they set $\beta' = 0$. Since n is indistinguishable, the adversary has to make a random guess here. Thus, the advantage $\mathbf{Adv}^{ddh}_{\mathcal{G},k,Gen()}(\mathcal{A})$ of the adversary to win the game is a negligible function.

A.4 Proof of Main Theorem

We utilize the IND-CPA framework (Definition 4 and 5) to demonstrate the unrecoverability of the original data ℓ. We devise a game denoted as $\boldsymbol{G}(\mathcal{A})$, depicted in Table 9. $\boldsymbol{G}(\mathcal{A})$ comprises four sub-games, namely, $\boldsymbol{G}_0(\mathcal{A})$, $\boldsymbol{G}_1(\mathcal{A})$, $\boldsymbol{G}_2(\mathcal{A})$, and $\boldsymbol{G}_3(\mathcal{A})$. There is only one distinction between every two consecutive sub-games, highlighted by the red rectangles, and the green rectangles indicate the sole difference between $\boldsymbol{G}_0(\mathcal{A})$ and $\boldsymbol{G}_3(\mathcal{A})$. Each sub-game comprises three processes: *Proc Initialize* (k, ℓ, F), *Proc LR* (ℓ, ℓ'), and *Proc Finalize* (β').

A.4.1 Game Settings

Proc Initialize(k, ℓ, F). In *Proc Initialize* (k, ℓ, F), all four sub-games generate the real M and return (G, γ, g, X). *Proc LR* (ℓ, ℓ'), where LR stands for Left-Right, denoting a procedure aimed at responding to queries posed by an adversary during the game [42]. In *Proc LR* (ℓ, ℓ'), each of the four sub-games supplies the adversary with an additional ℓ', referred to as a challenge message [29], which aids in facilitating the determination of whether ℓ is the original data.

[4] $g^{xy} = g^{xy \bmod p}$ by Lagrange's theorem.

Table 9: Code-based game-playing $\boldsymbol{G}(\mathcal{A})$ to prove $\mathscr{d}$ is indistinguishable.

Game $\boldsymbol{G}_0(\mathcal{A})$	**Game** $\boldsymbol{G}_1(\mathcal{A})$	**Game** $\boldsymbol{G}_2(\mathcal{A})$	**Game** $\boldsymbol{G}_3(\mathcal{A})$
Proc Initialize $(k,\mathscr{d},F)$	Proc Initialize $(k,\mathscr{d},F)$	Proc Initialize $(k,\mathscr{d},F)$	Proc Initialize $(k,\mathscr{d},F)$
$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$	$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$	$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$	$(G,p) \xleftarrow{R} \mathcal{G}(1^k)$
$g \xleftarrow{R} G^*$; $N \xleftarrow{R} \mathbb{Z}^{\gg n}$	$g \xleftarrow{R} G^*$; $N \xleftarrow{R} \mathbb{Z}^{\gg n}$	$g \xleftarrow{R} G^*$; $N \xleftarrow{R} \mathbb{Z}^{\gg n}$	$g \xleftarrow{R} G^*$; $N \xleftarrow{R} \mathbb{Z}^{\gg n}$
$\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$
$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$
$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$
$z \xleftarrow{R} xy \bmod p; n \xleftarrow{R} g^z$	$z \xleftarrow{R} xy \bmod p; n \xleftarrow{R} g^z$	$z \xleftarrow{R} xy \bmod p; n \xleftarrow{R} g^z$	$z \xleftarrow{R} xy \bmod p; n \xleftarrow{R} g^z$
$M \xleftarrow{R} Gen(\mathscr{d},n,N,\gamma,F)$	$M \xleftarrow{R} Gen(\mathscr{d},n,N,\gamma,F)$	$M \xleftarrow{R} Gen(\mathscr{d},n,N,\gamma,F)$	$M \xleftarrow{R} Gen(\mathscr{d},n,N,\gamma,F)$
Return (G,γ,g,X)	*Return* (G,γ,g,X)	*Return* (G,γ,g,X)	*Return* (G,γ,g,X)
Proc LR $(\mathscr{d},\mathscr{d}')$	Proc LR $(\mathscr{d},\mathscr{d}')$	Proc LR $(\mathscr{d},\mathscr{d}')$	Proc LR $(\mathscr{d},\mathscr{d}')$
$N \xleftarrow{R} \mathbb{Z}^{\gg n}$; $\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$N \xleftarrow{R} \mathbb{Z}^{\gg n}$; $\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$N \xleftarrow{R} \mathbb{Z}^{\gg n}$; $\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$	$N \xleftarrow{R} \mathbb{Z}^{\gg n}$; $\gamma \xleftarrow{R} \mathbb{Z}^{+<n+\frac{N}{4}-\rho}$
$y \xleftarrow{R} \mathbb{Z}_p^*$; $C_1 \leftarrow g^y$; $n' \leftarrow X^y$	$w \xleftarrow{R} \mathbb{Z}_p^*$; $C_1 \leftarrow g^w$; $n' \leftarrow X^w$	$w \xleftarrow{R} \mathbb{Z}_p^*$; $C_1 \leftarrow g^w$; $n' \leftarrow X^w$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $C_1 \leftarrow g^y$; $n' \leftarrow X^y$
$C_2 \xleftarrow{R} Gen(\mathscr{d},n',N,\gamma,F)$	$C_2 \xleftarrow{R} Gen(\mathscr{d},n',N,\gamma,F)$	$C_2 \xleftarrow{R} Gen(\mathscr{d}',n',N,\gamma,F)$	$C_2 \xleftarrow{R} Gen(\mathscr{d}',n',N,\gamma,F)$
Proc Finalize (β')	Proc Finalize (β')	Proc Finalize (β')	Proc Finalize (β')
Guess:	Guess:	Guess:	Guess:
(1). if $\begin{cases} C_1 = Y, & \mathcal{P}(C_1 = Y) = 0.5 \\ C_1 \neq Y, & \mathcal{P}(C_1 \neq Y) = 0.5 \end{cases}$	(1). if $\begin{cases} C_1 = Y, & \mathcal{P}(C_1 = Y) = 0.5 \\ C_1 \neq Y, & \mathcal{P}(C_1 \neq Y) = 0.5 \end{cases}$	(1). if $\begin{cases} C_1 = Y, & \mathcal{P}(C_1 = Y) = 0.5 \\ C_1 \neq Y, & \mathcal{P}(C_1 \neq Y) = 0.5 \end{cases}$	(1). if $\begin{cases} C_1 = Y, & \mathcal{P}(C_1 = Y) = 0.5 \\ C_1 \neq Y, & \mathcal{P}(C_1 \neq Y) = 0.5 \end{cases}$
(2). if $\begin{cases} C_2 = M, & \beta' = 1 \\ C_2 \neq M, & \beta' = 0 \end{cases}$	(2). if $\begin{cases} C_2 = M, & \beta' = 1 \\ C_2 \neq M, & \beta' = 0 \end{cases}$	(2). if $\begin{cases} C_2 = M, & \beta' = 1 \\ C_2 \neq M, & \beta' = 0 \end{cases}$	(2). if $\begin{cases} C_2 = M, & \beta' = 1 \\ C_2 \neq M, & \beta' = 0 \end{cases}$
Return (β')	*Return* (β')	*Return* (β')	*Return* (β')

Proc LR($\mathscr{d}$, $\mathscr{d}'$). During *Proc LR*($\mathscr{d}$, $\mathscr{d}'$), in $\boldsymbol{G}_0(\mathcal{A})$, the y is generated by $y \xleftarrow{R} \mathbb{Z}_p^*$, which is the same as the y generated in *Proc Initialize*(k, $\mathscr{d}$, F). Notably, generating y is an intermediate step in the derivation of n. Different y values yield distinct z values, consequently leading to different n. In $\boldsymbol{G}_1(\mathcal{A})$, distinguished from $\boldsymbol{G}_0(\mathcal{A})$, we generate w, a value distinct from y, by $w \xleftarrow{R} \mathbb{Z}_p^*$. As a result, $C1$ and n' differ between $\boldsymbol{G}_0(\mathcal{A})$ and $\boldsymbol{G}_1(\mathcal{A})$, following $C_1 \leftarrow g^y$, $n' \leftarrow X^y$ in $\boldsymbol{G}_0(\mathcal{A})$ and $C_1 \leftarrow g^w$, $n' \leftarrow X^w$ in $\boldsymbol{G}_1(\mathcal{A})$ respectively. It is evident that in $\boldsymbol{G}_0(\mathcal{A})$, $C_1 = Y$ and $n' = n$, whereas in $\boldsymbol{G}_1(\mathcal{A})$, $C_1 \neq Y$ and $n' \neq n$. When examining $\boldsymbol{G}_1(\mathcal{A})$ and $\boldsymbol{G}_2(\mathcal{A})$, the sole difference lies in C_2 generated by *Proc LR*($\mathscr{d}$, $\mathscr{d}'$). In $\boldsymbol{G}_1(\mathcal{A})$, we utilize the original data $\mathscr{d}$ as a parameter of $Gen()$ to generate C_2, i.e., $C_2 \xleftarrow{R} Gen(\mathscr{d},n',N,\gamma,F)$, but in $\boldsymbol{G}_2(\mathcal{A})$, C_2 is generated by the challenge message $\mathscr{d}'$, i.e., $C_2 \xleftarrow{R} Gen(\mathscr{d}',n',N,\gamma,F)$. Therefore, the C_2's in $\boldsymbol{G}_1(\mathcal{A})$ and $\boldsymbol{G}_2(\mathcal{A})$ are different. The difference between $\boldsymbol{G}_2(\mathcal{A})$ and $\boldsymbol{G}_3(\mathcal{A})$ is that we replace $w \xleftarrow{R} \mathbb{Z}_p^*$ in $\boldsymbol{G}_2(\mathcal{A})$ with $y \xleftarrow{R} \mathbb{Z}_p^*$ in $\boldsymbol{G}_3(\mathcal{A})$. In $\boldsymbol{G}_3(\mathcal{A})$, the y in *Proc LR*($\mathscr{d}$, $\mathscr{d}'$) is the same as the y in *Proc Initialize*(k, $\mathscr{d}$, F), so $C_1 = Y$ and $n' = n$. After understanding the differences between two neighboring games, we examine the distinction between $\boldsymbol{G}_0(\mathcal{A})$ and $\boldsymbol{G}_3(\mathcal{A})$, illustrated by the green rectangle. The only difference noted is that in $\boldsymbol{G}_0(\mathcal{A})$, C_2 is generated by $C_2 \xleftarrow{R} Gen(\mathscr{d},n',N,\gamma,F)$, whereas in $\boldsymbol{G}_3(\mathcal{A})$, C_2 is generated by $C_2 \xleftarrow{R} Gen(\mathscr{d}',n',N,\gamma,F)$. This achieves the intended distinction in the game $\boldsymbol{G}(\mathcal{A})$, where whether or not the original data is used to generate C_2 becomes the sole difference. All four sub-games in *Proc LR*($\mathscr{d}$, $\mathscr{d}'$) return (C_1, C_2) before *Proc Finalize*(β').

Proc Finalize(β'). In *Proc Finalize*(β'), the adversary initially checks if $C_1 = Y$, followed by verifying if $C_2 = M$, resembling a conditional probability distribution. Given the returned (G, γ, g, X) in *Proc Initialize*(k, $\mathscr{d}$, F) for the four sub-games, the adversary lacks the information to deduce Y. Therefore, they cannot ascertain whether $C_1 = Y$ when provided with $y \xleftarrow{R} \mathbb{Z}p^*$ and $w \xleftarrow{R} \mathbb{Z}_p^*$. Consequently, in the four sub-games, the probability that the adversary to guess $C_1 = Y$ is almost equal to the probability that they guess $C_1 \neq Y$, meaning a random guess. Subsequently, in determining whether $C_2 = M$, there are two scenarios. One is where the adversary assumes $C_1 = Y$ and believes $C_2 = M$, and the other is where they assume $C_1 \neq Y$ and then believe $C_2 = M$.

A.4.2 Proof

According to the conditional probability distribution, we can express the probability of the adversary guessing $C_2 = M$ in the four sub-games as from Eq. 19 to Eq. 22.

Game $\boldsymbol{G}_0(\mathcal{A})$. We start with Eq. 19.

$$\begin{aligned} \mathcal{P}(G_0(\mathcal{A}) = 1) = \mathcal{P}(C_1 = Y) \times \mathcal{P}\left(\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A}) = 1\right) \\ \times \mathcal{P}\left(\boldsymbol{Exp}_{\mathcal{G},k,Gen()}^{ddh-0}(\mathcal{A}) = 1\right) + \mathcal{P}(C_1 \neq Y) \\ \times \mathcal{P}\left(\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-1}(\mathcal{A}) = 1\right) \times \mathcal{P}\left(\boldsymbol{Exp}_{\mathcal{G},k,Gen()}^{ddh-0}(\mathcal{A}) = 1\right) \end{aligned} \tag{19}$$

With the premise that $\mathcal{P}(C_1 = Y)$ is a random guess, the probability that the adversary believes $n' = n$ can be considered as the game $\boldsymbol{Exp}_{\mathcal{G},k}^{ddh-0}(\mathcal{A})$, because if $C_1 = Y$, n' is equal to n. Similarly, the adversary guessing $C_2 = M$ can be considered as $\boldsymbol{Exp}_{\mathcal{G},k,Gen()}^{ddh-0}(\mathcal{A})$, given that M is indistinguishable

(Theorem 2) and the adversary is provided with the real ℓ in $\boldsymbol{G}_0(\mathcal{A})$. The success of the adversary's guess depends on whether $n = n'$, making it a conditional probability. Thus, the first case that the probability the adversary returns $\beta' = 1$ can be expressed as $\mathcal{P}(C_1 = Y) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)$.

The second case is similar to the first one, with the difference that the adversary considers $C_1 \neq Y$. With this premise, the adversary believes $n' \neq n$, which can be considered as the game $\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A})$. Thus, the second case that the probability that adversary returns $\beta' = 1$ can be expressed as $\mathcal{P}(C_1 \neq Y) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)$. Finally incorporating the two cases, we obtain the probability of the adversary returning $\beta = 1$ shown as Eq. 19.

Game $\boldsymbol{G}_1(\mathcal{A})$. The adversary's guess that $C_2 = M$ in $\boldsymbol{G}_1(\mathcal{A})$ can be descirbed in Eq. 20.

$$\begin{aligned}\mathcal{P}(G_1(\mathcal{A}) = 1) &= \mathcal{P}(C_1 = Y) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \\ &\times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right) + \mathcal{P}(C_1 \neq Y) \\ &\times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)\end{aligned} \tag{20}$$

The analysis of game $\boldsymbol{G}_1(\mathcal{A})$ is similar to $\boldsymbol{G}_0(\mathcal{A})$. The only difference is that w is no longer the value y in *Proc Intitialize(k, ℓ, F)*, leading to $C_1 \neq n$, which can be considered as the game $\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A})$.

Game $\boldsymbol{G}_2(\mathcal{A})$. The probability of the adversary guessing $C_2 = M$ in $\boldsymbol{G}_2(\mathcal{A})$ is described in Eq. 21. The analysis closely resembles that of game $\boldsymbol{G}_1(\mathcal{A})$ except for the utilization of ℓ' instead of ℓ in *Proc LR*(ℓ, ℓ'). Therefore, we refrain from reiterating the explanation.

$$\begin{aligned}\mathcal{P}(G_2(\mathcal{A}) = 1) &= \mathcal{P}(C_1 = Y) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \\ &\times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right) + \mathcal{P}(C_1 \neq Y) \\ &\times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)\end{aligned} \tag{21}$$

Game $\boldsymbol{G}_3(\mathcal{A})$. We formulate the probability of the adversary guessing $C_2 = M$ in $\boldsymbol{G}_3(\mathcal{A})$ as presented in Eq. 22.

$$\begin{aligned}\mathcal{P}(G_3(\mathcal{A}) = 1) &= \mathcal{P}(C_1 = Y) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \\ &\times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right) + \mathcal{P}(C_1 \neq Y) \\ &\times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right) \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)\end{aligned} \tag{22}$$

In $\boldsymbol{G}_3(\mathcal{A})$, we set $y \xleftarrow{R} \mathbb{Z}^*_p$ again in *Proc LR*(ℓ, ℓ') identical to the settings in $\boldsymbol{G}_0(\mathcal{A})$. Consequently, the task of determining whether $C_2 = M$ can be regarded as $\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A})$. Following the integration of the two cases discussed in $\boldsymbol{G}_0(\mathcal{A})$, we have Eq. 22.

Advantage to win $\boldsymbol{G}(\mathcal{A})$. As there is only one difference between the neighboring sub-games, so we consider the two neighboring sub-games as $\boldsymbol{Exp}^{ddh}_{\mathcal{G},k}(\mathcal{A})$ (Section A.3). The advantage of the adversary $\mathbf{Adv}^{ind-cpa}_{\mathcal{G},k,Gen()}(\mathcal{A})$ to win each $\boldsymbol{Exp}^{ddh}_{\mathcal{G},k}(\mathcal{A})$ can be expressed as from Eq. 23 to Eq. 25. Specifically, for $\boldsymbol{G}_0(\mathcal{A})$ and $\boldsymbol{G}_1(\mathcal{A})$, we have

$$\begin{aligned}&|\mathcal{P}(G_0(\mathcal{A}) = 1) - \mathcal{P}(G_1(\mathcal{A}) = 1)| \\ &= |\,0.5 \times \left[\mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) + \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right)\right] \\ &\quad \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right) - 0.5 \\ &\quad \times \left[\mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) + \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right)\right] \\ &\quad \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)| \\ &= |\mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right) - \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)| \\ &= \mathbf{Adv}^{ddh}_{\mathcal{G},k,Gen()} \leq \varepsilon\end{aligned} \tag{23}$$

For $\boldsymbol{G}_1(\mathcal{A})$ and $\boldsymbol{G}_2(\mathcal{A})$, based on Eq. 20 and Eq. 21, we have

$$|\mathcal{P}(G_1(\mathcal{A}) = 1) - \mathcal{P}(G_2(\mathcal{A}) = 1)| = 0 \leq \varepsilon \tag{24}$$

For $\boldsymbol{G}_2(\mathcal{A})$ and $\boldsymbol{G}_3(\mathcal{A})$, we can derive

$$\begin{aligned}&|\mathcal{P}(G_2(\mathcal{A}) = 1) - \mathcal{P}(G_3(\mathcal{A}) = 1)| \\ &= |\,0.5 \times \left[\mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) + \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right)\right] \\ &\quad \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right) \\ &\quad - 0.5 \times \left[\mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k}(\mathcal{A}) = 1\right) + \mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k}(\mathcal{A}) = 1\right)\right] \\ &\quad \times \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)| = |\mathcal{P}\left(\boldsymbol{Exp}^{ddh-1}_{\mathcal{G},k,Gen_0}(\mathcal{A}) = 1\right) \\ &\quad - \mathcal{P}\left(\boldsymbol{Exp}^{ddh-0}_{\mathcal{G},k,Gen()}(\mathcal{A}) = 1\right)| \\ &= \mathbf{Adv}^{ddh}_{\mathcal{G},k,Gen()} \leq \varepsilon\end{aligned} \tag{25}$$

In each $\boldsymbol{Exp}^{ddh}_{\mathcal{G},k}(\mathcal{A})$, the adversary's advantage is a negligible function. Summarizing Eq. 23, 24, and 25 based on the triangle inequality theorem, we can obtain Eq. 26, which is the advantage of the adversary to win the game $\boldsymbol{G}(\mathcal{A})$.

$$\begin{aligned}&\mathbf{Adv}^{ind-cpa}_{\mathcal{G},k,Gen()}(\mathcal{A}) = |\mathcal{P}(G_0(\mathcal{A}) = 1) - \mathcal{P}(G_3(\mathcal{A}) = 1)| \\ &\leq |\mathcal{P}(G_0(\mathcal{A}) = 1) - \mathcal{P}(G_1(\mathcal{A}) = 1)| \\ &\quad + |\mathcal{P}(G_1(\mathcal{A}) = 1) - \mathcal{P}(G_2(\mathcal{A}) = 1)| \\ &\quad + |\mathcal{P}(G_2(\mathcal{A}) = 1) - \mathcal{P}(G_3(\mathcal{A}) = 1)| \leq \varepsilon\end{aligned} \tag{26}$$

$\mathbf{Adv}^{ind-cpa}_{\mathcal{G},k,Gen()}(\mathcal{A})$ is a negligible function for PTAs. Therefore, ℓ is unrecoverable under PTAs.