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

Shuofeng Liu and Zihan Wang, The University of Queensland and CSIRO’s Data61; Minhui Xue, CSIRO’s Data61; Long Wang and Yuanchao Zhang, Information Security Department, Ant Group, MYBank; Guangdong Bai, The University of Queensland

https://www.usenix.org/conference/usenixsecurity24/presentation/liu-shuofeng

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.
August 14–16, 2024 • Philadelphia, PA, USA
978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.
Being Transparent is Merely the Beginning: 
Enforcing Purpose Limitation with Polynomial Approximation

Shuofeng Liu\textsuperscript{1,2} Zihan Wang\textsuperscript{1,2} Minhui Xue\textsuperscript{2} Long Wang\textsuperscript{3} Yuanchao Zhang\textsuperscript{3} Guangdong Bai\textsuperscript{1}

\textsuperscript{1}The University of Queensland \textsuperscript{2}CSIRO’s Data61 \textsuperscript{3}Information Security Department, Ant Group, MYBank

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\textsuperscript{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

\textsuperscript{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.
Data Owner Authorization

(a)

Scenario #1

(b)

Scenario #2

Legend:

- Data Owner
- App Client
- Website
- Processing Server
- Third Party

Data Owner Authorization

Figure 1: Two typical application scenarios of ALGO·SPEC.

Figure 2: A comparison between the work models of ALGO·SPEC 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 ALGO·SPEC. To position ALGO·SPEC, 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

ALGO·SPEC 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 ALGO·SPEC 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 ALGO·SPEC. Figure 3 illustrates its general workflow, and the detailed
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.

\[
P(x) = \alpha_0 x^n + \alpha_{n-1} x^{n-1} + \ldots + \alpha_2 x^2 + \alpha_1 x + \beta,
\]

where \(x\) is the variable, \(\alpha_i x^i\)’s are the non-constant terms, \(\alpha_i\)’s are coefficients, \(\beta\) 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. 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

<table>
<thead>
<tr>
<th>Notations</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D)</td>
<td>original dataset</td>
</tr>
<tr>
<td>(D^{(i)})</td>
<td>value of (D) at (i)th row, (j)th column</td>
</tr>
<tr>
<td>(d)</td>
<td>unspecified value in dataset</td>
</tr>
<tr>
<td>(F_x)</td>
<td>authorized algorithm</td>
</tr>
<tr>
<td>(F)</td>
<td>polynomial approximation of (F_x)</td>
</tr>
<tr>
<td>(n)</td>
<td>highest order of (F)</td>
</tr>
<tr>
<td>(F_n)</td>
<td>malicious algorithms</td>
</tr>
<tr>
<td>(M)</td>
<td>obscured list</td>
</tr>
<tr>
<td>(N)</td>
<td>the number of elements in (M)</td>
</tr>
<tr>
<td>(A)</td>
<td>calculation rule</td>
</tr>
<tr>
<td>(d)</td>
<td>exponent adjustment factor</td>
</tr>
<tr>
<td>(\lambda)</td>
<td>coefficient adjustment factor</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>the number of distances</td>
</tr>
<tr>
<td>(r)</td>
<td>distance list</td>
</tr>
<tr>
<td>(R)</td>
<td>result by ALGOSPEC</td>
</tr>
<tr>
<td>(G)</td>
<td>cyclic group</td>
</tr>
<tr>
<td>(G_{\langle})</td>
<td>cyclic group generated by (G)</td>
</tr>
<tr>
<td>(g)</td>
<td>random generator</td>
</tr>
<tr>
<td>(S)</td>
<td>possibility</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Setup (Data Owner)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data owner authorizes data processor to collect data with the desired purpose disclosed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setup (Data Processor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data processor obtains authorization for data collection, and discloses the desired algorithm (F_x) to be applied to the collected data.</td>
</tr>
<tr>
<td>Data processor collects original data from data owner.</td>
</tr>
<tr>
<td>Data processor determines (n) of the polynomial.</td>
</tr>
<tr>
<td>Data processor converts (F_x) to polynomial (F) by polynomial approximation.</td>
</tr>
<tr>
<td>Data processor acquires several components from ALGOSPEC, including (d) (exponent adjustment factor), (\lambda) (coefficient adjustment factor), (\gamma) (the number of distances), (pairs) (the pairs selected following distances), and (times) (times that pairs are used).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setup (Third Party)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third party sets up execution environment.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALGOSPEC generates obscured list (M) using (D), (F), (d), (\lambda), (\gamma).</td>
</tr>
<tr>
<td>When (M) is generated, calculation rule (A) is generated automatically.</td>
</tr>
<tr>
<td>Data processor sends (M) and (A) to third party.</td>
</tr>
<tr>
<td>Third party computes the result (r) through (M) and (A).</td>
</tr>
<tr>
<td>Third party sends (r) to data processor.</td>
</tr>
</tbody>
</table>

Figure 3: Overall workflow of ALGOSPEC.
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., $\beta$, 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 $\lambda_i$, 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 $\lambda$ and $d$. Below we explain how each $m$ in $M'$ is constructed.

For $m_1$ in $M'$, ALGOSPEC sets $m_1 \leftarrow \frac{\alpha_1}{\lambda_1} x^{d_1}$, meaning that the first non-constant term depends on $\lambda_1$ and $d_1$. Specifically, we divide the coefficient $\alpha_1$ by the corresponding adjustment factor $\lambda_1$, and subtract the order of $n \times x$ from the value of the corresponding position $d_1$ in the exponent adjustment vector. For $m_2$ in $M'$, we set $m_2 \leftarrow \lambda_n x^{d_n}$ with the same $\lambda_n$ 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 $m_{2n-1} \leftarrow \frac{\alpha_{2n-1}}{\lambda_{2n-1}} x^{d_{2n-1}}$ and $m_{2n} \leftarrow \lambda_{2n} x^{d_{2n}}$. 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 $\beta$. By computing $m_1 \times m_2 + ... + m_{2n-1} \times m_{2n}$, the result being equal to the calculation of $\alpha_0 x^0 + \alpha_{n-1} x^{n-1} + ... + \alpha_1 x$ can be derived, i.e.,

$$m_1 \times m_2 + ... + m_{2n-1} \times m_{2n} = \alpha_0 x^0 + ... + \alpha_1 x.$$  

(2)

The two vectors $\lambda$ and $d$ mainly serve to construct the two factors of a term, while also introducing randomness into the distribution of $m_s$. An extremely large or small $\lambda$ 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 $m_s$. They are adjustable in the actual deployment of ALGOSPEC.

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

#### 3.2.2 Processing of constant term $\beta$

ALGOSPEC decomposes $\beta$ 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 $\beta$ as Eq. 3, i.e.,

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

To ensure that the final computation result matches the desired $\beta$, 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 $m_s$ 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 $\Gamma$. The number of distances in $\Gamma$ is denoted as $\tau$. Following each distance in $\Gamma$, it builds a list $\Theta$ containing $\tau$ pairs, i.e., $(\theta_1, \theta_2), ..., (\theta_{2\tau-1}, \theta_{2\tau})$.

**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 $\Gamma$, it adds new distances such that all distances end up with an occurrence number close to $K$. To achieve this, $(m_{2n+1}, m_{2n+2}), ..., (m_{N-3}, m_{N-2})$ is generated, where each $m_s$ follows Gaussian distribution, i.e., $m_s \sim N(0, \theta_i)$.

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

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

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

\(^2\)The number of new distances should not be less than $p$ 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 \((m_{2n+1}, m_{2n+2}), \ldots, (m_{N-1}, m_N)\), 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

\[
m_1 \times m_2 \times \cdots \times m_{2n-1} \times m_{2n} + \cdots + m_{N-1} \times m_N = \alpha_x x^p + \cdots + \beta.
\]

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 original result \(R\) when the original authorized algorithm is executed directly, i.e., \(r = R + \eta\), where \(\eta\) is a negligible value signifying the margin of error.

\[
r = m_1 \times m_2 \times \cdots \times m_{2n-1} \times m_{2n} + \cdots + m_{N-1} \times m_N.
\]

Here, the data processor can delegate part of the complex calculation of \(M\) to the third party. Specifically, for those \(m_i\)s 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.

\[
x_{ij}' = \frac{x_{ij} - \min(x_{ij})}{\max(x_{ij}) - \min(x_{ij})};
\]

\[
p_{ij} = \frac{x_{ij}'}{\sum_{i=1}^{m} x_{ij}'};
\]

\[
e_j = -\frac{1}{\ln(n)} \sum_{i=1}^{m} p_{ij} \ln(p_{ij});
\]

\[
W_j = \frac{1 - e_j}{\sum_{j=1}^{n}(1 - e_j)}.
\]

where \(i\) represents the \(i^{th}\) sample, \(j\) represents the \(j^{th}\) feature and \(n\) 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 \(M_i\)s to the third party and meanwhile requests them to proceed with the remaining calculations following the calculation rule A. Upon receiving \(M_i\)s 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

\[
p(x_i | y) = \prod_{j=1}^{d_f} p(x_{ij} | y),
\]

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 \(\mu_k\) denote the mean and \(\sigma_k^2\) denote the Bessel corrected variance of the values in \(x\) associated with class \(C_k\), respectively. That is,

\[
\mu_k = \frac{1}{n_k} \sum_{x_i \in C_k} x_i,
\]

where \(n_k\) represents the number of samples of class \(C_k\), and

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

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

\[
p(x = v | C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v - \mu_k)^2}{2\sigma_k^2}}.
\]
Finally, the category $pred$ with the highest probability of occurrence of new sample $v$ is selected, i.e.,

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

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

When receiving the $M$s, the third party computes $\sigma_k$ 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 $M$s to represent $(v - \mu_k)^2$ of each feature and sends them to the third party again. Once the third party receives the $M$s, $p(x = v|C_k)$ can be calculated by $\sigma_k$ 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 $\text{Adv}^{ddh}_{G,k} (\mathcal{A})$ in solving the DDH problem relative to $G$ can be defined as the difference between the probability that $\text{Exp}^{ddh-0}_{G,k} (\mathcal{A})$ outputs $\text{true}$ and the probability that $\text{Exp}^{ddh-1}_{G,k} (\mathcal{A})$ outputs $\text{true}$, i.e.,

$$\text{Adv}^{ddh}_{G,k} (\mathcal{A}) = |\mathbb{P}(\text{Exp}^{ddh-0}_{G,k} (\mathcal{A}) = \text{true}) - \mathbb{P}(\text{Exp}^{ddh-1}_{G,k} (\mathcal{A}) = \text{true})|.  \tag{16}$$

In the DDH problem, a finite cyclic group $G = <G>$ is generated by a generator $g$, and a random generator $g \leftarrow G$ as well as random integers $a, b \leftarrow \mathbb{Z}_n^*$ 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 defined at random from $\mathbb{Z}_n^*$ [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 $<G>$, given $x = g^a$ and $y = g^b$, the DLP is to calculate $g^{ab \mod p}$ 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 ciphertext 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, $\mathbb{P} (\text{Enc}(m_0) = c) - \mathbb{P} (\text{Enc}(m_1) = c) \leq \varepsilon$ holds, where $m_0$ and $m_1$ are two different plaintexts, $c$ is the ciphertext produced by encrypting either one of them and $\varepsilon$ is a negligible function.

The most common type of indistinguishability is the indistinguishability under chosen-plaintext attack (IND-CPA),...
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 algo-
rithm 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 $n^h$-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 can-
not 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 $n_2$, respec-
tively, $M_1$ 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 $\delta$). Under a CPA, the
original data $\delta$ 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 $\delta$,
and generate another challenge $C'$ in the last sub-game using
a randomly generated $\delta'$. By demonstrating that consecutive
sub-games satisfy $|P(G_i(\mathcal{A}) = 1) - P(G_{i+1}(\mathcal{A}) = 1)| \leq \varepsilon$, ($i \in [0, 2]$), the first and last sub-games also satisfy
$|P(G_0(\mathcal{A}) = 1) - P(G_3(\mathcal{A}) = 1)| \leq \varepsilon$. The detailed proof
is presented in Appendix A.4.

5.3  Discussion on $n$’s Confidentiality

We have proved that $\delta$ is unrecoverable when $n$ is sufficiently
large through code-based game-playing. However, even when
$n$ is not exceptionally large, the protection for $\delta$ can still be
ensured. The only way for the adversary to get $\delta$ 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 $n$s they selected, which means they face a plethora of
NP-hard problems. Therefore, a relatively small $n$ is suffi-
cient 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 compre-

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Precision</th>
<th>Apply</th>
<th>Expanded</th>
</tr>
</thead>
<tbody>
<tr>
<td>WH</td>
<td>149</td>
<td>3-digit</td>
<td>EM</td>
<td></td>
</tr>
<tr>
<td>WH ep1</td>
<td>$10^3$</td>
<td>3-digit</td>
<td>EM</td>
<td>✓</td>
</tr>
<tr>
<td>WH ep2</td>
<td>$10^4$</td>
<td>3-digit</td>
<td>EM</td>
<td>✓</td>
</tr>
<tr>
<td>WH ep3</td>
<td>$10^5$</td>
<td>3-digit</td>
<td>EM</td>
<td>✓</td>
</tr>
<tr>
<td>WH ep4</td>
<td>$10^6$</td>
<td>3-digit</td>
<td>EM</td>
<td>✓</td>
</tr>
<tr>
<td>breast cancer</td>
<td>569</td>
<td>6-digit</td>
<td>NB</td>
<td></td>
</tr>
<tr>
<td>heart disease</td>
<td>1,025</td>
<td>single-prec.</td>
<td>NB</td>
<td></td>
</tr>
<tr>
<td>hypertension</td>
<td>26,058</td>
<td>single-prec.</td>
<td>NB</td>
<td></td>
</tr>
<tr>
<td>hypertension ep</td>
<td>104,232</td>
<td>single-prec.</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Datasets used in this paper.

Datasets. We mainly use the breast cancer [1], heart dis-
edisease [2], 2021 world happiness report [5], and hyperten-
sion [10] datasets to assess the efficacy of ALGOSPEC. To
provide a thorough evaluation of our methodology across var-
ied 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.
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-HexA N [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 $\frac{1}{2}$ 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 $\epsilon_{dp}$ and sensitivity to 1.0; for Gaussian mechanism, we set $\epsilon_{dp}$, $\delta_{dp}$, 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 $10^3$ on entropy method and $10^6$ 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 $10^2$ to $10^6$ on entropy method.

<table>
<thead>
<tr>
<th>PT</th>
<th>ET</th>
<th>TT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.000</td>
<td>0.235</td>
</tr>
<tr>
<td>DP (L)</td>
<td>1.007</td>
<td>0.218</td>
</tr>
<tr>
<td>DP (G)</td>
<td>1.030</td>
<td>0.219</td>
</tr>
<tr>
<td>FHE</td>
<td>0.134</td>
<td>487.663</td>
</tr>
<tr>
<td>ALGOSPEC</td>
<td>36.378</td>
<td>1.069</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PT</th>
<th>ET</th>
<th>TT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.000</td>
<td>0.235</td>
</tr>
<tr>
<td>DP (L)</td>
<td>1.007</td>
<td>0.218</td>
</tr>
<tr>
<td>DP (G)</td>
<td>1.030</td>
<td>0.219</td>
</tr>
<tr>
<td>FHE</td>
<td>0.134</td>
<td>487.663</td>
</tr>
<tr>
<td>ALGOSPEC</td>
<td>36.378</td>
<td>1.069</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Heart Disease ($10^2$)</th>
<th>Breast Cancer ($10^2$)</th>
<th>Hypertension ($10^4$)</th>
<th>Hypertension ep ($10^5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>F1-score</td>
<td>Accuracy</td>
<td>F1-score</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.844</td>
<td>0.851</td>
<td>0.921</td>
</tr>
<tr>
<td>DP (L)</td>
<td>0.698</td>
<td>0.705</td>
<td>0.904</td>
</tr>
<tr>
<td>DP (G)</td>
<td>0.790</td>
<td>0.786</td>
<td>0.895</td>
</tr>
<tr>
<td>FHE</td>
<td>0.473</td>
<td>0.460</td>
<td>0.465</td>
</tr>
<tr>
<td>ALGOSPEC</td>
<td>0.824</td>
<td>0.833</td>
<td>0.921</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Heart Disease ($10^4$)</th>
<th>Breast Cancer ($10^2$)</th>
<th>Hypertension ($10^5$)</th>
<th>Hypertension ep ($10^5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT ET TT</td>
<td>PT ET TT</td>
<td>PT ET TT</td>
<td>PT ET TT</td>
</tr>
<tr>
<td>Baseline</td>
<td>0.000 0.513 0.513</td>
<td>0.000 0.467 0.467</td>
<td>0.000 2.735 2.735</td>
</tr>
<tr>
<td>DP (L)</td>
<td>0.082 0.575 0.657</td>
<td>0.104 0.544 0.648</td>
<td>2.034 2.362 4.396</td>
</tr>
<tr>
<td>DP (G)</td>
<td>0.059 0.576 0.635</td>
<td>0.077 0.545 0.622</td>
<td>1.421 2.316 3.737</td>
</tr>
<tr>
<td>FHE</td>
<td>0.040 9.980 10.020</td>
<td>0.269 3.097 3.366</td>
<td>0.243 196.428 196.672</td>
</tr>
<tr>
<td>ALGOSEC</td>
<td>1.432 1.520 2.952</td>
<td>2.011 1.645 3.656</td>
<td>35.915 41.415 77.330</td>
</tr>
</tbody>
</table>

**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, ALGOSEC 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.

ALGOSEC 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 ALGOSEC’s generalization for simple and complex algorithms, as well as the challenges of deploying ALGOSEC in practice.

**Generalization.** Given the target application scenarios of ALGOSEC, the algorithms are typically not simplistic. ALGOSEC 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 ALGOSEC 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 ALGOSEC for parameterized algorithms as future work.

**Deployment.** When deploying ALGOSEC 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 ALGOSEC 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., \( \varepsilon_{dp} \) and \( \delta_{dp} \), as well as sensitivity. In general, the larger the values of \( \varepsilon_{dp} \) and \( \delta_{dp} \), 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, \( \varepsilon_{dp} \), and \( \delta_{dp} \) 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 two main 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

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


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

Lemma 1. \( n \) is initialized indistinguishable. Let \( \text{Exp}^{\ddh}(\mathcal{A}) \) be a code-based game-playing. \( \text{Exp}^{\ddh-0}(\mathcal{A}) \) and \( \text{Exp}^{\ddh-1}(\mathcal{A}) \) are the sub-games in \( \text{Exp}^{\ddh}(\mathcal{A}) \), which is illustrated in Table 7. In Proc Initialize (k) of both sub-games, the only difference is \( z \overset{R}{\leftarrow} \mathbb{Z}_p^* \) in \( \text{Exp}^{\ddh-1}(\mathcal{A}) \). In proc Finalize (b'), the adversary guesses if \( Z = g^{\log_p x + y} \) and returns \( b' = 1 \) if the answer is yes, otherwise \( b' = 0 \) in the two sub-games. We have that \( n \) is indistinguishable if

\[
\text{Adv}^{\ddh} (\mathcal{A}) = |\mathbb{P} (\text{Exp}^{\ddh-0}(\mathcal{A}) = 1) - \mathbb{P} (\text{Exp}^{\ddh-1}(\mathcal{A}) = 1)| \leq \varepsilon,
\]

where \( \varepsilon \) 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 \( G = \langle G \rangle \) be the finite cyclic group; \( p \) is the prime order; \( k = |p| \) be the security parameter; \( g \overset{R}{\leftarrow} G^\ast \) be the generated element. We generate \( n \) by \( x \overset{R}{\leftarrow} \mathbb{Z}_p^* \); \( y \overset{R}{\leftarrow} \mathbb{Z}_p^* \); \( z \overset{R}{\leftarrow} \mathbb{Z}_p^* \) \( mod \) \( p \Rightarrow n = g^z \).

To prove that \( n \) is indistinguishable, we construct a code-based game-playing \( \text{Exp}^{\ddh-0}(\mathcal{A}) \) and game \( \text{Exp}^{\ddh-1}(\mathcal{A}) \). Each sub-game contains two processes, the initialization process \( \text{Proc Initialize} (k) \) and finalization process \( \text{Proc Finalize} (b') \), where \( b' \) is a flag that can be set to 1 or 0 based on the adversary guess.

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

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

Lemma 2. \( n \) is unrecovable from \( M \). Let \( F \) be a polynomial with the degree \( n \). Let \( M \) denote a list containing \( N \) pairs \((N \gg 2n)\) with almost evenly distribution, which follows \( \gamma \) distances. \( \gamma \) is well-designed by \( \gamma = \rho + \omega \), where \( \rho \) is the number of distances of \( n \) pairs and \( \omega \) is the new distances to be created \((0 \leq \omega \leq \frac{N}{2} - \rho)\). Let \( A \) denote the calculation rule in \( \text{ALGOSPEC} \). The CPA is unable to discern \( n \) of the polynomial when provided with \( M \).
Table 8: Code-based game-playing $\text{Exp}^{\text{ddh}}_{G,k,\text{Gen}()}(\mathcal{A})$ to prove $M$ is indistinguishable

| Game $\text{Exp}^{\text{ddh}-0}_{G,k,\text{Gen}()}(\mathcal{A})$: |
|-----------------|-----------------|
| Proc Initialize $(k, d, F)$: | Proc Finalize $(\gamma)$: |
| $(G, p) \leftarrow G(1^k)$ | $\gamma \leftarrow Z_n$ |
| $g \leftarrow G^\gamma$; $N \leftarrow Z^{\geq n}$ | $\gamma \leftarrow Z^{< n}$ |
| $y \leftarrow Z_p$ | $y \leftarrow Z_p$ |
| $M \leftarrow \text{Gen}(d, n, N, \gamma, F)$ | $M \leftarrow \text{Gen}(d, n, N, \gamma, F)$ |
| Return $M$ | Return $M$ |
| Guess: | Guess: |
| if $n = g^y$, set $\beta' = 1$ | if $n = g^y$, set $\beta' = 1$ |
| $n \neq g^y$, set $\beta' = 0$ | $n \neq g^y$, set $\beta' = 0$ |
| Return $(\beta')$ | Return $(\beta')$ |

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 $\gamma$ (almost evenly distributed) in the claim with the same $\rho$ and $\omega$. Subsequently, we establish the correctness of Lemma 2 by demonstrating the validity of Claim 1.

Claim 1. Given $n$ pairs following $\rho$ distances and $(\frac{N}{2} - n)$ 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 $\gamma$ 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 $\lambda$ and $d$ to adjust the number of distances, ensuring an almost uniform distribution within each distance. $M$ comprises $\frac{N}{2}$ pairs distributed across $\gamma$ patterns. Considering $\rho \in [1, n]$ and $\omega \in [0, \frac{N}{2} - \rho]$, it follows that $\gamma \in [1, n + \frac{N}{2} - \rho]$. 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 $\gamma \in (n, n + \frac{N}{2} - \rho)$, 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 $\text{Exp}^{\text{ddh}}_{G,k,\text{Gen}()}(\mathcal{A})$ used in our proof. Its construction follows the intuition that two Ms generated using different $n$ are indistinguishable. $\text{Exp}^{\text{ddh}-0}_{G,k,\text{Gen}()}(\mathcal{A})$ and $\text{Exp}^{\text{ddh}-1}_{G,k,\text{Gen}()}(\mathcal{A})$ are the sub-games. In $\text{Exp}^{\text{ddh}-0}_{G,k,\text{Gen}()}(\mathcal{A})$, $n$ is set by $n \leftarrow R \times \mathbb{Z}_p$; in $\text{Exp}^{\text{ddh}-1}_{G,k,\text{Gen}()}(\mathcal{A})$, $n$ is set by $n \leftarrow R \times \mathbb{Z}_p$. Both sub-games return $M$ with their corresponding $n$ in Proc Initialize$(k, d, F)$. The adversary is not able to distinguish which $M$ is generated from the real $n$. $M$ is indistinguishable if

$$\text{Adv}^{\text{ddh}}_{G,k,\text{Gen}()}(\mathcal{A}) = |\beta \left(\text{Exp}^{\text{ddh}-0}_{G,k,\text{Gen}()}(\mathcal{A}) = 1\right) - \beta \left(\text{Exp}^{\text{ddh}-1}_{G,k,\text{Gen}()}(\mathcal{A}) = 1\right)| \leq \epsilon$$

The Proc Initialize$(k, d, F)$ of the two sub-games is designed to generate $M$. Their only difference is marked with red rectangles. In $\text{Exp}^{\text{ddh}-0}_{G,k,\text{Gen}()}(\mathcal{A})$, $n$ is the real one generated by $n \leftarrow R \times \mathbb{Z}_p$, whereas in $\text{Exp}^{\text{ddh}-1}_{G,k,\text{Gen}()}(\mathcal{A})$, $n$ is generated randomly. In the Proc Finalize$(\beta)$, 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 $n = g^{\gamma y}$ or not based on the returned $Ms$. If they believe $n = g^{\gamma y}$, 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 $\text{Adv}^{\text{ddh}}_{G,k,\text{Gen}()}(\mathcal{A})$ 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 $d$. We devise a game denoted as $G(\mathcal{A})$, depicted in Table 9. $G(\mathcal{A})$ comprises four sub-games, namely, $G_0(\mathcal{A})$, $G_1(\mathcal{A})$, $G_2(\mathcal{A})$, and $G_3(\mathcal{A})$. 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 $G_0(\mathcal{A})$ and $G_3(\mathcal{A})$. Each sub-game comprises three processes: Proc Initialize$(k, d, F)$, Proc LR$(d, d')$, and Proc Finalize$(\beta')$.

A.4.1 Game Settings

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

\[ g^{\gamma y} = g^{\gamma y} \pmod{p} \text{ by Lagrange’s theorem.} \]
Table 9: Code-based game-playing $G(\mathcal{A})$ to prove $d$ is indistinguishable.

<table>
<thead>
<tr>
<th>Game $G_0(\mathcal{A})$</th>
<th>Game $G_1(\mathcal{A})$</th>
<th>Game $G_2(\mathcal{A})$</th>
<th>Game $G_3(\mathcal{A})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proc Initialize $(k, d, F)$</td>
<td>Proc Initialize $(k, d, F)$</td>
<td>Proc Initialize $(k, d, F)$</td>
<td>Proc Initialize $(k, d, F)$</td>
</tr>
<tr>
<td>$(G, p) \xrightarrow{$} G {1}$</td>
<td>$g \xrightarrow{$} G'$; $N \xrightarrow{$} Z^{\kappa}$</td>
<td>$g \xrightarrow{$} G'$; $N \xrightarrow{$} Z^{\kappa}$</td>
<td>$g \xrightarrow{$} G'$; $N \xrightarrow{$} Z^{\kappa}$</td>
</tr>
<tr>
<td>$x \xrightarrow{$} Z_p$; $X \leftarrow g^x$</td>
<td>$y \xrightarrow{$} Z_p$; $Y \leftarrow g^y$</td>
<td>$z \xrightarrow{$} xy \text{ mod } p; n \xrightarrow{$} g^z$</td>
<td>$M \xrightarrow{$} Gen(d, n, \gamma, F)$</td>
</tr>
<tr>
<td>$\text{Return} (G, \gamma, g, X)$</td>
<td>$\text{Return} (G, \gamma, g, X)$</td>
<td>$\text{Return} (G, \gamma, g, X)$</td>
<td>$\text{Return} (G, \gamma, g, X)$</td>
</tr>
</tbody>
</table>

Proc LR$(d, d')$. During Proc LR$(d, d')$, in $G_0(\mathcal{A})$, the $y$ is generated by $y \xrightarrow{\$} Z_p$, which is the same as the $y$ generated in Proc Initialize$(k, d, F)$. 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 $G_1(\mathcal{A})$, distinguished from $G_0(\mathcal{A})$, we generate $w$, a value distinct from $y$, by $w \xrightarrow{\$} Z_p$. As a result, $C_1$ and $n'$ differ between $G_0(\mathcal{A})$ and $G_1(\mathcal{A})$, following $C_1 \leftarrow g^w$, $n' \leftarrow X'$ in $G_0(\mathcal{A})$ and $C_1 \leftarrow g^w$, $n' \leftarrow X'$ in $G_1(\mathcal{A})$ respectively. It is evident that in $G_0(\mathcal{A})$, $C_1 = Y$ and $n' = n$, whereas in $G_1(\mathcal{A})$, $C_1 \neq Y$ and $n' \neq n$. When examining $G_1(\mathcal{A})$ and $G_2(\mathcal{A})$, the sole difference lies in $C_2$ generated by Proc LR$(d, d')$. In $G_1(\mathcal{A})$, we utilize the original data $d$ as a parameter of $Gen()$ to generate $C_2$, i.e., $C_2 \xrightarrow{\$} Gen(d, n', \gamma, F)$, but in $G_2(\mathcal{A})$, $C_2$ is generated by the challenge message $d'$, i.e., $C_2 \xrightarrow{\$} Gen(d', n', \gamma, F)$. Therefore, the $C_2$’s in $G_1(\mathcal{A})$ and $G_2(\mathcal{A})$ are different. The difference between $G_2(\mathcal{A})$ and $G_3(\mathcal{A})$ is that we replace $w \xrightarrow{\$} Z_p$ in $G_2(\mathcal{A})$ with $y \xrightarrow{\$} Z_p$ in $G_3(\mathcal{A})$. In $G_3(\mathcal{A})$, the $y$ in Proc LR$(d, d')$ is the same as the $y$ in Proc Initialize$(k, d, F)$, so $C_1 = Y$ and $n' = n$. After understanding the differences between two neighboring games, we examine the distinction between $G_0(\mathcal{A})$ and $G_3(\mathcal{A})$, illustrated by the green rectangle. The only difference noted is that in $G_0(\mathcal{A})$, $C_2$ is generated by $C_2 \xrightarrow{\$} Gen(d, n', N, \gamma, F)$, whereas in $G_3(\mathcal{A})$, $C_2$ is generated by $C_2 \xrightarrow{\$} Gen(d', n', N, \gamma, F)$. This achieves the intended distinction in the game $G(\mathcal{A})$, where whether or not the original data is used to generate $C_2$ becomes the sole difference. All four sub-games in Proc LR$(d, d')$ return $(C_1, C_2)$ before Proc Finalize$(\beta')$.

Proc Finalize$(\beta')$. In Proc Finalize$(\beta')$, the adversary initially checks if $C_1 = Y$, followed by verifying if $C_2 = M$, resembling a conditional probability distribution. Given the returned $(G, \gamma, g, X)$ in Proc Initialize$(k, d, F)$ for the four sub-games, the adversary lacks the information to deduce $Y$. Therefore, they cannot ascertain whether $C_1 = Y$ when provided with $y \xrightarrow{\$} Z_p$ and $w \xrightarrow{\$} Z_p$. 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 $G_0(\mathcal{A})$. We start with Eq. 19.

$$D(G_0(\mathcal{A})) = 1 = D(C_1 = Y) \times P\left(Exp_{d_h}^{\rho_1}(\mathcal{A}) = 1\right)$$

$$\times P\left(Exp_{d_h-0}^{\rho_1}(\mathcal{A} = 1) \times D(C_1 \neq Y)\right) \quad (19)$$

With the premise that $D(C_1 = Y)$ is a random guess, the probability that the adversary believes $n' = n$ can be considered as the game $Exp_{d_h-0}^{\rho_1}(\mathcal{A})$, because if $C_1 = Y$, $n'$ is equal to $n$. Similarly, the adversary guessing $C_2 = M$ can be considered as $Exp_{d_h-0}^{\rho_1}(\mathcal{A})$, given that $M$ is indistinguishable.
(Theorem 2) and the adversary is provided with the real $\delta$ in $G_0(\mathcal{A})$. 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 $\Pr(C_1 = Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1)$. 

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 $\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A})$. Thus, the second case that the probability that adversary returns $\beta' = 1$ can be expressed as $\Pr(C_1 \neq Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1)$. Finally incorporating the two cases, we obtain the probability of the adversary returning $\beta = 1$ shown as Eq. 19. 

**Game $G_1(\mathcal{A})$.** The adversary’s guess that $C_2 = M$ in $G_1(\mathcal{A})$ can be described in Eq. 20.

\[
\Pr(G_1(\mathcal{A}) = 1) = \Pr(C_1 = Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(C_1 \neq Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \tag{20}
\]

The analysis of game $G_1(\mathcal{A})$ is similar to $G_0(\mathcal{A})$. The only difference is that $w$ is no longer the value $y$ in $\text{Proc \, Initialize}(k, \delta, F)$, leading to $C_1 \neq n$, which can be considered as the game $\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A})$.

**Game $G_2(\mathcal{A})$.** The probability of the adversary guessing $C_2 = M$ in $G_2(\mathcal{A})$ is described in Eq. 21. The analysis closely resembles that of game $G_1(\mathcal{A})$ except for the utilization of $\delta'$ instead of $\delta$ in $\text{Proc \, LR}(\delta, \delta')$. Therefore, we refrain from reiterating the explanation.

\[
\Pr(G_2(\mathcal{A}) = 1) = \Pr(C_1 = Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(C_1 \neq Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \tag{21}
\]

**Game $G_3(\mathcal{A})$.** We formulate the probability of the adversary guessing $C_2 = M$ in $G_3(\mathcal{A})$ as presented in Eq. 22.

\[
\Pr(G_3(\mathcal{A}) = 1) = \Pr(C_1 = Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(C_1 \neq Y) \times \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \times \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \tag{22}
\]

In $G_3(\mathcal{A})$, we set $y \leftarrow Z_n^*$ again in $\text{Proc \, LR}(\delta', \delta'')$ identical to the settings in $G_0(\mathcal{A})$. Consequently, the task of determining whether $C_2 = M$ can be regarded as $\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A})$. Following the integration of the two cases discussed in $G_0(\mathcal{A})$, we have Eq. 22.

**Advantage to win $G(\mathcal{A})$.** As there is only one difference between the neighboring sub-games, we consider the two neighboring sub-games as $\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A})$. The advantage of the adversary $\text{Adv}_{G, k, \text{Gen}}^{\text{ind-cpa}}(\mathcal{A})$ to win each $\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A})$ can be expressed as from Eq. 23 to Eq. 25. Specifically, for $G_0(\mathcal{A})$ and $G_1(\mathcal{A})$, we have

\[
|\Pr(G_0(\mathcal{A}) = 1) - \Pr(G_1(\mathcal{A}) = 1)| = 0.5 \times \left[ \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) + \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \right] - 0.5 \times \left[ \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) + \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \right] \tag{23}
\]

For $G_1(\mathcal{A})$ and $G_2(\mathcal{A})$, based on Eq. 20 and Eq. 21, we have

\[
|\Pr(G_1(\mathcal{A}) = 1) - \Pr(G_2(\mathcal{A}) = 1)| = 0 \leq \epsilon \tag{24}
\]

For $G_2(\mathcal{A})$ and $G_3(\mathcal{A})$, we can derive

\[
|\Pr(G_2(\mathcal{A}) = 1) - \Pr(G_3(\mathcal{A}) = 1)| = 0.5 \times \left[ \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) + \Pr(\text{Exp}_{G, k}^{\text{ddh}}(\mathcal{A}) = 1) \right] - 0.5 \times \left[ \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) + \Pr(\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A}) = 1) \right] \tag{25}
\]

\[
|\Pr(G_3(\mathcal{A}) = 1) - \Pr(G_3(\mathcal{A}) = 1)| = 0 \leq \epsilon \tag{26}
\]

In each $\text{Exp}_{G, k, \text{Gen}}^{\text{ddh}}(\mathcal{A})$, 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 $G(\mathcal{A})$.

\[
\text{Adv}_{G, k, \text{Gen}}^{\text{ind-cpa}}(\mathcal{A}) = |\Pr(G_0(\mathcal{A}) = 1) - \Pr(G_3(\mathcal{A}) = 1)| \leq |\Pr(G_0(\mathcal{A}) = 1) - \Pr(G_1(\mathcal{A}) = 1)| + |\Pr(G_1(\mathcal{A}) = 1) - \Pr(G_2(\mathcal{A}) = 1)| + |\Pr(G_2(\mathcal{A}) = 1) - \Pr(G_3(\mathcal{A}) = 1)| \leq \epsilon \tag{26}
\]

$\text{Adv}_{G, k, \text{Gen}}^{\text{ind-cpa}}(\mathcal{A})$ is a negligible function for PTAs. Therefore, $\delta$ is un recoverable under PTAs.