Article

Not peer-reviewed version

Outsourced Privacy-Preserving Feature Selection Based on Fully Homomorphic Encryption

Koki Wakiyama , Tomohiro I , Hiroshi Sakamoto [*]

Posted Date: 20 May 2025

doi: 10.20944/preprints202505.1442.v1

Keywords: privacy-preserving machine learning; fully homomorphic encryption; feature selection; outsourced computation

Article

Outsourced Privacy-Preserving Feature Selection Based on Fully Homomorphic Encryption

Koki Wakiyama, Tomohiro I and Hiroshi Sakamoto *

Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka, Japan
* Correspondence: hiroshi@ai.kyutech.ac.jp

Abstract: Feature selection is a technique that extracts a meaningful subset from a set of features in training data. When the training data is large-scale, appropriate feature selection enables the removal of redundant features, which can improve generalization performance, accelerate the training process, and enhance the interpretability of the model. This study proposes a privacy-preserving computation model for feature selection. Generally, when the data owner and analyst are the same, there is no need to conceal the private information. However, when they are different parties or when multiple owners exist, an appropriate privacy-preserving framework is required. Although various private feature selection algorithms, they all require two or more computing parties and do not guarantee security in environments where no external party can be fully trusted. To address this issue, we propose the first outsourcing algorithm for feature selection using fully homomorphic encryption. Compared to a prior two-party algorithm, our result improves the time and space complexity $O(kn^2)$ to $O(kn\log^3 n)$ and $O(kn)$, where k and n denote the number of features and data samples, respectively. We also implemented the proposed algorithm and conducted comparative experiments with the naive one. The experimental result shows the efficiency of our method even with small datasets.

Keywords: privacy-preserving machine learning; fully homomorphic encryption; feature selection; outsourced computation

1. Introduction

Feature selection is the process of identifying and selecting useful features (variables or attributes) from a dataset with the aim of improving the performance of predictive models and optimizing learning efficiency [1]. Consider a scenario in which a target variable is to be predicted using a large number of potential explanatory variables, with the goal of building a simple yet effective predictive model from a vast set of observations. For instance, in semiconductor manufacturing, more than tens of thousands of external factors can influence the production process from silicon material to the final product. If it is possible to identify in advance which of these factors are critical to product quality, then it is sufficient to construct a classification model based only on those selected features—offering a range of benefits. In practice, such preprocessing can lead to improvements in model accuracy, reductions in training time, and enhanced interpretability of the model. Many feature selection algorithms based on various criteria have been proposed to date [2–8].

In this study, we consider the feature selection in situations where the training data contains privacy-sensitive information. When the data owner who needs a reasonable learning model and the analyst required to construct the model are the same, privacy is generally not a concern as long as there is no data leakage. However, our focus is on privacy-preserving computation in cases where these two parties are separate. Existing frameworks to address this issue include techniques such as *secret sharing* [9] and *differential privacy* [10]. In this work, from the perspective of outsourcing, we focus on an approach based on *fully homomorphic encryption* (FHE) [11], which is a form of public-key cryptosystem.

In secret sharing schemes, data is split and distributed among parties, which introduces a risk of data leakage if those parties colludes maliciously. Differential privacy, on the other hand, prevents leakage by injecting random noise into the data. However, there exists a trade-off between the amount of noise added and the quality of the learning results, and it cannot guarantee complete protection against data leakage.

By contrast, FHE enables all arithmetic and comparison operations to be performed directly on encrypted data. Since only the data owner possessing the secret key can decrypt the final result, the privacy guarantees are as strong as the underlying public-key encryption. The primary limitations of FHE have been computational inefficiency and the challenge of outsourcing computation in scenarios involving multiple data owners. However, recent advancements such as TFHE [12] and multi-key homomorphic encryption schemes [13] are steadily overcoming these obstacles.

Therefore, we consider a simplified privacy-preserving problem for feature selection: The data owner retains the training data along with a pair of generated public and private keys. Then, the analyst receives the encrypted training data and performs feature selection using FHE scheme. The result of the computation is then returned to the data owner, who decrypts it using the private key.

Note that the proposed method constitutes a fully outsourced computation [14,15] with only a single round of communication between the data owner and the analyst. This aspect significantly strengthens privacy protection. In non-fully outsourced scenarios, multiple interactions or queries from a party may be required. During such interactions, partial decryption of maliciously crafted messages from untrusted analyst could lead to unintentional leakage of the data owner's private information.

To prevent this, conventional approaches typically rely on assumptions that constrain the behavior of the analysts. For example, the *semi-honest* model assumes that the analyst follows the protocol faithfully but may attempt to extract as much information as possible from the data he receives. However, imposing such assumptions compromises practical applicability. In contrast, our algorithm requires only single round communication without decryption and ensures that the risk of data leakage depends solely on the strength of the cryptosystem. The TFHE [12] used in this study is proven to be at least secure against chosen-plaintext attacks (IND-CPA secure). Consequently, it offers a significant advantage in terms of both security and practical usability compared to existing methods.

2. Related Works and Our Contributions

Feature selection is broadly categorized into filter method [16], wrapper method [17], and embedded method [18]. Among these, the filter method evaluates the importance of features based on information theory, making it faster than other methods and highly generalizable due to its independence from specific models. While evaluation metrics such as mutual information are commonly used in filter methods [19], this study focuses on feature selection based on a recently proposed *consistency measure* [20]. Feature selection algorithms using consistency measures have been shown to achieve both high computational efficiency and high classification accuracy on large-scale real-world datasets [7,8]. In contrast, feature selection algorithms based on wrapper or embedded methods are generally impractical for secure computation due to the high computational cost associated with searching for optimal feature subsets or training models. Therefore, this study proposes a consistency measure-based secure feature selection.

Homomorphic encryption (HE) is a framework for secure computation that leverages the homomorphic properties of public-key cryptosystems. Specifically, if encrypted integers can be added without decryption, the cryptosystem is said to be *additive*. Furthermore, if it supports both operations, it is called FHE, e.g., RSA [21], the first public-key cryptosystem, is multiplicative but not additive. Additionally, RSA is not a *probabilistic* encryption scheme—i.e., it does not produce different ciphertexts for the same plaintext in each execution—making it vulnerable to chosen-plaintext attacks. The first probabilistic HE [22] was capable of computing the sum (i.e., bitwise XOR) of encrypted bits. Subsequently, a scheme enabling the addition of integers was proposed [23]. Later, the first HE capable of both addition and multiplication was introduced [24], but it allowed only single multiplication operation in overall computation, limiting its practicality for secure computation. Eventually, the first FHE, which imposes no restrictions on the number of operations, was proposed [11], making arbitrary secure computations theoretically possible. In practice, fast FHE libraries have become available [25], expanding its applicability. The implementation of the algorithm proposed in this study also utilizes such a library.

Although several private feature selection methods have been proposed to date [26–29], only a few have addressed privacy-preserving computation for the consistency measure [30], and this study is the first to attempt a fully outsourced computation for it. This is because the computation of the optimal consistency measure reduces to the *set cover problem* known to be NP-hard [31]. Consequently, approximation algorithms and acceleration techniques for solving this optimization problem have been proposed [7,8], and in this study, we realize a secure fully outsourced computation of such an algorithm.

A core technique in this approximate method is preprocessing via sorting. By sorting the entire dataset with feature values as keys, it becomes possible to efficiently narrow down the important features, enabling a heuristic approach to approximate optimal feature selection. However, executing this algorithm over encrypted data is not straightforward. In general, the lower bound for comparison-based sorting algorithms is $\Omega(n \log n)$, but no secure outsourced computation method is currently known that achieves this lower bound. At present, the fastest known method uses sorting networks with a time complexity of $O(n \log^2 n)$ [33], which is also adopted in this study as preprocessing. It should be noted, however, that if secret sharing among two or more parties is used, faster privacy-preserving sorting becomes feasible [34]. Moreover, disregarding practical constraints, it is theoretically possible to construct sorting networks with $O(n \log n)$ time complexity [35]. Nonetheless, both of these approaches are outside the scope of this study.

Table 1 below summarizes HE-based private feature selection algorithms. Here, k and n denote the number of features and data samples, respectively. A filter method [26] adopts the χ^2 statistic as its evaluation criterion and employs additively homomorphic encryption [23]. While this method has a low offline computational cost, its communication overhead is greater than that of the proposed method. Moreover, since it assumes two-party computation, it does not support fully outsourced computation. Additionally, it has not been implemented. Another two-party protocol [30] uses the same consistency measure as the proposed method and adopts the fastest FHE [12]. Although it also achieves exactly single round complexity as our the method, it assumes a two-party setting and therefore does not support fully outsourced computation.

Table 1. Comparison of HE-based feature selection algorithms.

Algorithm	Metric	Time	Space	# Round	Outsourced
Rao et al. [26]	χ^2	$O(kn) \cdot \mathcal{C}_P$	$O(kn)$	$O(1)$	partially
Ono et al. [30]	consistency	$O(kn^2) \cdot \mathcal{C}_C$	$O(kn^2)$	1	partially
Proposed	consistency	$O(kn \log^3 n) \cdot \mathcal{C}_C$	$O(kn)$	1	fully

$\mathcal{C}_P$ and $\mathcal{C}_C$ are costs per single operation depending on the respective cryptosystems [12,23]. To enable a fair comparison under the same conditions, the computational complexity of Ono et al. [30] is described only with respect to feature selection task, omitting the preprocessing time required for data formatting. The term "partially" refers to decrypting the data in part before obtaining the final result.

3. Preliminaries

3.1. Consistency-Based Feature Selection

Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of indices of data, associated with a set $F = \{f_1, f_2, \ldots, f_k\}$ of features and a class variable C, where the feature value $d(f_i)$ and the class label $d(C)$ are defined for each data $d \in D$. Table 2 shows an example of the triple (D, F, C).

The feature selection is to find a minimal subset $F' \subseteq F$ relevant to C, where F' is said to be *consistent* if, for any $d, d' \in D$, $d(f_i) = d'(f_i)$ for all $f_i \in F'$ implies $d(C) = d'(C)$, and F' is minimal, if any proper subset of F' is no longer consistent.

For example, consider the mutual information $I(F'; C)$ for evaluating the relevance of F' showing in Table 2. We can see that f_1 is more important than f_5 due to the fact $I(f_1; C) > I(f_5; C)$. f_1 and f_2 of Table 2 will be chosen to explain C based on the score of I. However, a closer examination of D reveals that f_1 and f_2 cannot uniquely determine C. In fact, we find d_2 and d_5 with $d_2(f_1) = d_5(f_1)$ and $d_2(f_2) = d_5(f_2)$ but $d_2(C) \neq d_5(C)$. On the other hand, we can see that f_4 and f_5 uniquely determine C using the formula $C = f_4 \oplus f_5$ while $I(f_4; C) = I(f_5; C) = 0$. It becomes clear that $I(F'; C)$ alone may not always lead to appropriate feature selection.

Table 2. An example triple (D, F, C) shown in [8].

D	f_1	f_2	f_3	f_4	f_5	C
d_1	1	0	1	1	1	0
d_2	1	1	0	0	0	0
d_3	0	0	0	1	1	0
d_4	1	0	1	0	0	0
d_5	1	1	1	1	0	1
d_6	0	1	0	1	0	1
d_7	0	1	0	0	1	1
d_8	0	0	0	0	1	1
$I(f_i; C)$	0.189	0.189	0.049	0.000	0.000	

Let us review the notion of the consistency measure employed in this study. A consistency measure $\mu : 2^F \to [0, \infty)$ is a function to represent how far the data deviate from a consistent state, where F is required to satisfy *determinisity* (i.e., $\mu(F) = 0$ iif F is consistent) and *monotonicity* (i.e., $F \subseteq G$ implies $\mu(F) \geq \mu(G)$). In this study, we focus on the binary consistency: $\mu_{\text{bin}}(F) = 0$, F is consistent; 1, otherwise [7]. For other consistency measures, see e.g., rough set [2], ICR [5], and inconsistent pair [6].

As Table 2 illustrates, the importance of a feature f_i is not determined solely by itself, but is influenced by the relative relationship among other features. CWC [8] achieves superior feature selection compared to other statistical measures by computing the relative importance of features. Algorithm 1 outlines the procedure of CWC, where the consistency of the candidate set excluding f_i is evaluated to determine whether to select f_i. Here, the order in which features f_i are selected significantly affects the result, so it is important to preprocess the triple (D, F, C) in advance. For example, there is a known method that determines the order of f_i by sorting the triple with the values of f_i as keys. This study also adopts that method.

Algorithm 1 CWC [7] over plaintexts

```
Require: A dataset $(D, F, C)$;
Ensure: A minimal consistent subset of $F$;
1: for $i = 1, \dots, k$ do
2:   if $F \setminus \{f_i\}$ is consistent then
3:     update $F \leftarrow F \setminus \{f_i\}$;
4:   end if
5: end for
6: return $F$;
```

3.2. Computation on FHE

The proposed FHE-based private feature selection is implemented using TFHE [25], one of the fastest FHE library for bitwise addition (XOR '$\oplus$') and bitwise multiplication (AND '$\cdot$') over ciphertext. On TFHE, any ℓ-bit integer $m = (m_1, m_2, \dots, m_\ell)$ is encrypted bitwise: $E[m] \equiv (E[m_1], E[m_2], \dots, E[m_\ell])$.

Given $E[b]$ and $E[b']$ ($b, b' \in \{0, 1\}$), FHE allows to compute the bitwise operations $E[b \oplus b']$ and $E[b \cdot b']$ without decrypting $E[b]$ and $E[b']$. It also allows all arithmetic and logical operations via XOR and AND as follows: Let x, y represent ℓ-bit integers and x_i, y_i the i-th bit of x, y, respectively. Let c_i represent the i-th carry-in bit and s_i the i-th bit of the sum $x + y$. Then, we can obtain the private full-adder $E[x + y]$ using the relations $s_i = x_i \oplus y_i \oplus c_i$ and $c_{i+1} = (x_i \oplus c_i) \cdot (y_i \oplus c_i) \oplus c_i$. We can construct other operations such as subtraction, multiplication, and division based on the full-adder.

On the other hand, we can also obtain the private comparison $E[cmp(>, x, y)]$ satisfying $cmp(>, x, y) = 1$ if $x > y$ and 0 otherwise, because $cmp(>, x, y)$ is identical to the most significant bit of $y + (-x)$, which can be obtained without decryption. Here, $(-x)$ is the bit complement of x obtained by $x_i \oplus 1$ for all i-th bits.

Thus, by using FHE, it is possible to perform all arithmetic operations and comparisons for encrypted variables or elements of arrays. For simplicity, we represent such operations in plaintext notation throughout the remainder of this paper. That is, unless otherwise stated or unless it may cause confusion, an operation such as the addition of two encrypted integers $E[x]$ and $E[y]$, resulting in $E[x + y]$, will be denoted simply as $x + y$.

Based on FHE, we can design a private sorting algorithm as follows (e.g. [32]). Note that, in this code, the variables `gt`, `tmp`, and any entry in the array `arr[1..n]` are all encrypted. Therefore, after this code is executed, the resulting `arr[1..n]` is sorted securely without revealing the private information about `arr[1..n]`.

```
// private bubble_sort over ciphertexts
void bubble_sort(int *arr,int n){
    for(int i=0;i<n-1;i++){
        for(int j=1;j<n-1;j++){
            int gt=cmp(>,arr[j-1],arr[j]);
            int tmp=gt*arr[j-1]^(!gt)*arr[j];
            arr[j-1]=gt*arr[j]^(!gt)*arr[j-1];
            arr[j]=tmp;
        }
    }
}
```

However, it is difficult to improve the $O(n^2)$ time algorithm to $O(n \log n)$. This is because encrypting pointers is fundamentally meaningless. If pointers are in plaintext, information about the input (e.g., the distribution of values) may be leaked. On the other hand, if the pointers are encrypted, it becomes impossible to access the corresponding memory addresses, and thus computation cannot proceed. Satisfying these conflicting requirements simultaneously appears to be impossible. This difficulty can be partially avoided by using *sorting network*, where comparisons are performed only between fixed pairs of elements, eliminating the need for pointers. Theoretically, we can construct an optimum sorting network that achieves $O(n \log n)$ comparisons [35], but the circuit size is impractically large. Therefore, in practice, $O(n \log^2 n)$ algorithm [33] is used. This study also adopts this approach.

4. Method

4.1. Sorting Network on FHE

Sorting via FHE is typically performed using sorting networks. It remains an open question whether a fastest algorithm such as quicksort or mergesort can be effectively realized in FHE setting. In this study, we adopt the Batcher's odd-even mergesort [33] as the basis for sorting. Batcher's algorithm employs a recursively constructed sorting network according to the number of elements and achieves sorting in $O(n \log^2 n)$ time. However, it requires that the number of data items be a power of two. Therefore, if the input data size does not meet this condition, we insert appropriate dummy elements beforehand to adjust the size. Moreover, since sorting network is basically not a stable sort, we attach a $\log n$-bit suffix to each data element to ensure stability. This preprocessing does not affect the correctness or quality of feature selection in any way.

4.2. A Naive Algorithm for Private CWC

In this study, we assume that the data analyst (i.e., an algorithm) receives the input triple (D, F, C) in encrypted form, and that the algorithm is capable of performing FHE operations on encrypted data.

Here, (D, F, C) is possessed as an array, and the algorithm has access to any encrypted entry $d_i(f_j)$ or $d_i(C)$ for $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, k\}$. The private CWC (denoted pCWC) utilizes sorting D with the feature vectors $\vec{F} = (f_1, f_2, \ldots, f_k)$ formed by $f_i \in F$. By sorting (D, F, C) with $\vec{F}$ as the key, identical feature vectors are placed adjacent to each other, significantly speeding up the consistency checking step.

Table 3 (upper) shows an input (D, F, C) and Table 3 (lower) shows the sorted (D, F, C) where each original index d_i is renamed by the sorted order D_j and for each $i \in \{1, 2, \ldots, k\}$, the corresponding feature label $L[i][j]$ is the unique grouping label of $\log n$ bits for D_j defined by the feature vector $\vec{F}[..i] = (f_1, f_2, \ldots, f_i)$.

For example, Table 3 (lower), the labels L_5 for D_1 and D_2 are both 000, indicating that D_1 and D_2 belong to the same group based on identical feature vector values up to f_5. Therefore, the consistency check needs to be performed only for data points that share the same label associated with the tail f_i. Thus, by using the sorted (D, F, C) along with the corresponding feature labels in $L[1..k][1..n]$, the consistency checking process becomes significantly simplified.

Table 3. Sorted (D, F, C) and corresponding feature labels L_i.

D	f_1	f_2	f_3	f_4	f_5	C
d_1	1	0	1	0	0	1
d_2	0	1	0	0	1	0
d_3	0	1	0	0	1	0
d_4	1	0	0	0	1	1
d_5	1	0	1	1	1	0
sorted D	f_1 : $L[1][1..5]$	f_2 : $L[2][1..5]$	f_3 : $L[3][1..5]$	f_4 : $L[4][1..5]$	f_5 : $L[5][1..5]$	C
$D_1(= d_2)$	0:000	1:000	0:000	0:000	1:000	0
$D_2(= d_3)$	0:000	1:000	0:000	0:000	1:000	0
$D_3(= d_4)$	1:001	0:001	0:001	0:001	1:001	1
$D_4(= d_1)$	1:001	0:001	1:010	0:010	0:010	1
$D_5(= d_5)$	1:001	0:001	1:010	1:011	1:011	0

Algorithm 2 shows a naive pCWC computation based on the sorted (D, F, C) and the feature labels. Given (D, F, C), consider the evaluation of each feature f_t in $F = \{f_1, f_2, \ldots, f_k\}$. The algorithm evaluates the consistency on the feature set $F' = F \setminus \{f_t\}$, obtained by removing f_t, and obtains the corresponding Boolean value b_t. Specifically, b_t is computed as:

$$b_t = \bigwedge_{i=2}^{n} ((D_{i-1} \neq D_i) \vee (D_{i-1}(C) = D_i(C)))$$

Here, $b_t = 1$ iff f_t is selected. By reporting the resulting $\vec{b} = (b_1, \ldots, b_k)$ to the data owner, the selected features are obtained after decrypting $\vec{b}$.

We next analyze the complexity of our algorithm where in the following discussion, we assume that the cost of each homomorphic operation under FHE is constant. That is, the computational cost of arithmetic and comparison operations on encrypted data is treated as $O(1)$, unless otherwise specified.

Algorithm 2 Naive pCWC for ciphertexts

```
Require: An encrypted $(D, F, C)$;
Ensure: A minimal consistent subset of $F$;
1: for $t = k, k-1, \ldots, 1$ do
2:     sort $(D, F', C)$ with $\vec{F}'$ as the key for $F' = F \setminus \{f_t\}$;
3:     compute $L[j][i]$ for all $j = 1, 2, \ldots, k$ and $i = 1, 2, \ldots, n$;
4:     compute the consistency $b_t \in \{0, 1\}$ of $(D, F', C)$;
5:     update $D_i(f_t) \leftarrow D_i(f_t) \cdot b_t$ for all $i = 1, 2, \ldots, n$;
6: end for
7: return $\vec{b} = (b_1, b_2, \ldots, b_k)$ representing the selected subset of $F$;
```

Theorem 1. *Algorithm 2 (naive pCWC) on ciphertexts simulates Algorithm 1 (CWC) on plaintexts. The time and space complexities are $O(k^2 n \log^3 n)$ and $O(kn)$ for $|F| = k$ and $|D| = n$, respectively.*

Proof. Assuming that (D, F', C) is already sorted, the feature label $L[t][i]$ of D_i for $\vec{F}[..t]$ can be computed based on whether $D_{i-1}(..f_t) = D_i(..f_t)$ holds, where $D_i(..f_t)$ stands for $D_i(f_1) \cdots D_i(f_t)$. When the predecessor $L[t][i-1]$ for D_{i-1} is already defined, using the logical bit x to indicate whether $D_{i-1}(..f_t) = D_i(..f_t)$, the next $L[t][i]$ is defined as: $L[i] \leftarrow L[i-1] + \neg x$. These computations can be performed on FHE. Moreover, the logical bit b_t, which determines whether the feature f_t should be selected, can also be computed by FHE according to its definition. By updating each f_t as: $D_i(f_t) \leftarrow D_i(f_t) \cdot b_t$, we can effectively remove f_t from (D, F, C). Thus, Algorithm 2 correctly simulates the original CWC on plaintexts.

When sorting the encrypted (D, F', C) using a sorting network, the number of comparisons required is $O(kn \log^2 n)$. Since our method appends a $\lceil \log n \rceil$-bit suffix to each data entry to achieve stable sorting, the computation time for one feature f_t becomes $O(kn \log^3 n)$. Therefore, the total computation time across all features is $O(k^2 n \log^3 n)$. The space complexity is clear. □

4.3. Improved Private CWC

In the naive approach, sorting of (D, F', C) is iterated for each $F' = F \setminus \{f_t\}$ with specified $f_t \in F$, resulting in high computational cost. Therefore, we aim to accelerate this algorithm. It should be noted that, since (D, F, C) are encrypted, pointer-based referencing cannot be used. In this study, we propose a method that avoids full sorting at each step by partitioning F into two sequences—selected features and unprocessed features—and processing them independently before merging the results.

The key concept here is to utilize the feature labels in $L[t][1..n]$ introduced in Algorithm 2 for partial sorting. Suppose that we are focusing on a current feature f_t. That is, feature selection has already been completed for the suffix vector $\vec{F}[t+1..] = (f_{t+1}, f_t, \ldots, f_k)$. In this case, we define the feature labels for $\vec{F}[t+1..]$ and store them in the array $PostL[1..n]$ as follows: $PostL[i] = PostL[i']$ iff $D_i(f_{j+1}..) = D_{i'}(f_{j+1}..)$. This groups the suffix sequences of F that share the same feature vector.

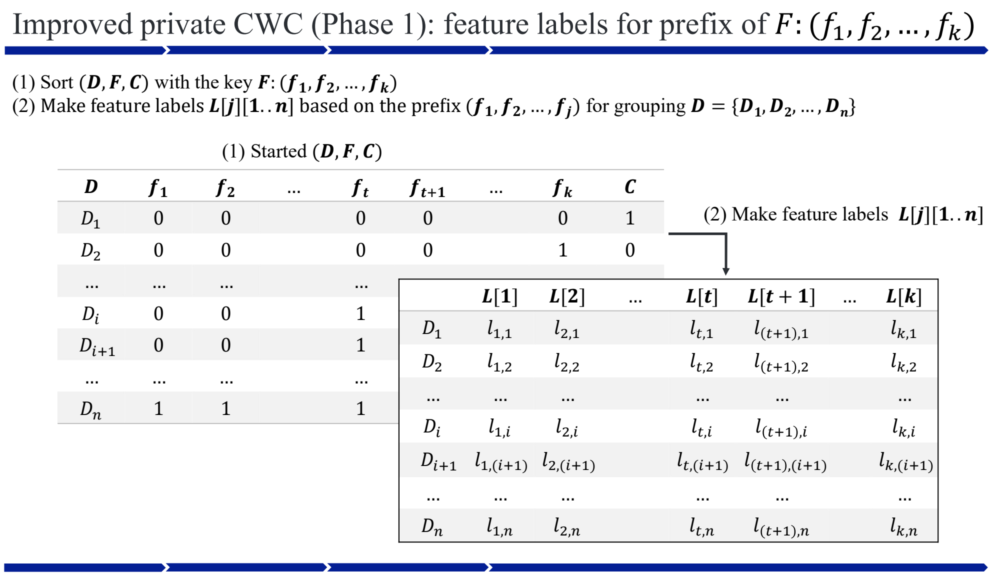
As shown in Algorithm 2, feature labels for any prefix vector $\vec{F}[..t]$ are stored in the array $L[t][1..n]$. Therefore, to group $(D, F \setminus \{f_t\}, C)$, we first independently compute $L[t][1..n]$ for prefix vector $\vec{F}[..t]$ and $PostL[1..n]$ for the corresponding suffix vector $\vec{F}[t+1..]$, and then calculate the ranks of $L[t][1..n]$ and $PostL[1..n]$ to merge them correctly. This approach enables us to avoid iterative sorting for the entire (D, F', C).

Algorithm 3 Improved pCWC for ciphertexts

```
Require: An encrypted $(D, F, C)$;
Ensure: A minimal consistent subset of $F$;
 1: sort $(D, F, C)$ with $\vec{F}$ as the key;
 2: compute the feature label $L[j][i]$ for all $j = 1, 2, \ldots, k$ and $i = 1, 2, \ldots, n$;
 3: initialize $PostL[1..n] \leftarrow (0, 0, \ldots, 0)$ and $MapL[1..n] \leftarrow (1, 2, \ldots, n)$;
 4: for $t = k, \ldots, 1$ do
 5:     sort $(f_{t+1}[1..n], PostL[1..n], C[1..n], MapL[1..n])$ as the key $f_{t+1}[1..n]$;
 6:     update $PostL[1..n]$ by the sorted $f_{t+1}[1..n]$;
 7:     sort $(f_t, L[t-1][1..n])$ by the inverse $MapL^{-1}[1..n]$;
 8:     compute the consistency $b_t \in \{0, 1\}$ of $(L[t-1][1..n], PostL[1..n], C[1..n])$;
 9:     update $f_t : D_\ell(f_t) \leftarrow D_\ell(f_t) \cdot b_t$ for all $\ell = 1, 2, \ldots, n$;
10: end for
11: return $\vec{b} = (b_1, b_2, \ldots, b_k)$ representing the selected subset of $F$;
```

The task of Algorithm 3 is divided into three phases: preprocessing on (D, F, C) (Figure 1), partial sorting of (D, F, C) with respect to a specific feature f_t (Figure 2), and the decision-making and update of (D, F, C) based on f_t (Figure 3), respectively.

Figure 1 illustrates the preprocessing of (D, F, C). Here, it is assumed that the sorting of (D, F, C) has already been completed, and each D_i represents its rank in the sorted order. For this sorted (D, F, C), a label L_t is assigned to each D_i based on $\vec{F}[..t]$, such that $L[t][i] = L[t][j]$ iff $D_i(..f_t) = D_j(..f_t)$. That is, $L[t][1..n]$ group labels for D_is sharing the same value of $\vec{F}[..t]$.


Improved private CWC (Phase 1): feature labels for prefix of F: $(f_1, f_2, \ldots, f_k)$
(1) Sort $(\boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C})$ with the key $\boldsymbol{F}$: $(\boldsymbol{f_1}, \boldsymbol{f_2}, \ldots, \boldsymbol{f_k})$
(2) Make feature labels $\boldsymbol{L[j][1..n]}$ based on the prefix $(\boldsymbol{f_1}, \boldsymbol{f_2}, \ldots, \boldsymbol{f_j})$ for grouping $\boldsymbol{D} = \{\boldsymbol{D_1}, \boldsymbol{D_2}, \ldots, \boldsymbol{D_n}\}$
(1) Started $(\boldsymbol{D}, \boldsymbol{F}, \boldsymbol{C})$
(2) Make feature labels $\boldsymbol{L[j][1..n]}$

D	f_1	f_2	...	f_t	f_{t+1}	...	f_k	C
D_1	0	0		0	0		0	1
D_2	0	0		0	0		1	0
...	...	...		...				
D_i	0	0		1				
D_{i+1}	0	0		1				
...	...	...		...				
D_n	1	1		1				

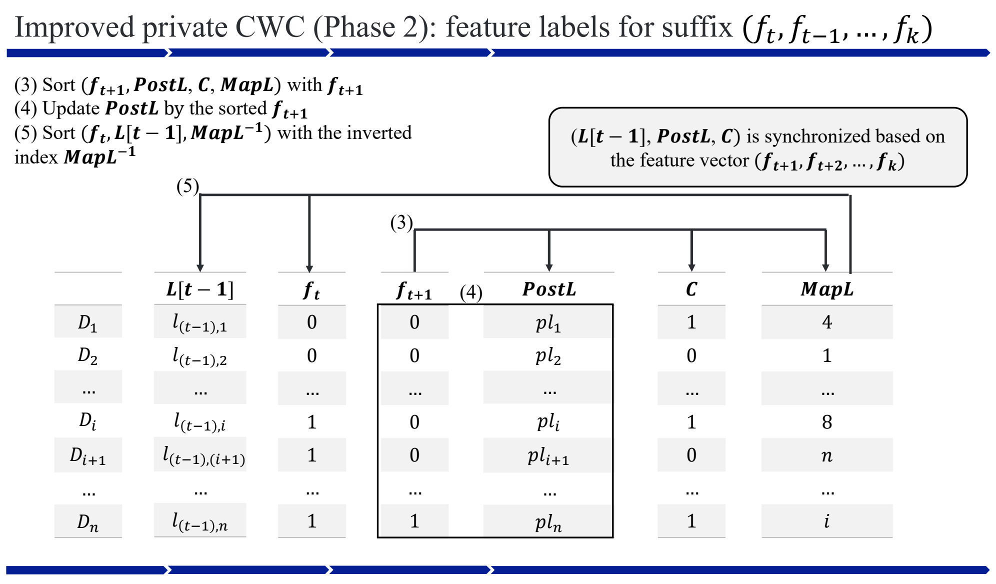
	$L[1]$	$L[2]$	...	$L[t]$	$L[t+1]$	...	$L[k]$
D_1	$l_{1,1}$	$l_{2,1}$		$l_{t,1}$	$l_{(t+1),1}$		$l_{k,1}$
D_2	$l_{1,2}$	$l_{2,2}$		$l_{t,2}$	$l_{(t+1),2}$		$l_{k,2}$
...	...	...		...	...		...
D_i	$l_{1,i}$	$l_{2,i}$		$l_{t,i}$	$l_{(t+1),i}$		$l_{k,i}$
D_{i+1}	$l_{1,(i+1)}$	$l_{2,(i+1)}$		$l_{t,(i+1)}$	$l_{(t+1),(i+1)}$		$l_{k,(i+1)}$
...	...	...		...	...		...
D_n	$l_{1,n}$	$l_{2,n}$		$l_{t,n}$	$l_{(t+1),n}$		$l_{k,n}$

Figure 1. Phase 1 of the improved algorithm corresponding to Line 1 - 3 in Algorithm 3.

Next, we focus on a specific feature f_t and determine whether it should be selected. For this purpose, we require the sorting result of (D, F', C), where $F' = F \setminus \{f_t\}$ (see Figure 2). However, since $\vec{F}[..t-1]$ has already been sorted, we divide F into $\vec{F}[..t-1]$, f_t, and $\vec{F}[t+1..]$ with the current f_t, and only $\vec{F}[t+1..]$ is sorted . Because the sorting of $\vec{F}[..t-1]$ is performed sequentially for $t = k, k-1, \ldots, t$, it only needs to be executed for the current value of f_t. As a result, the computational cost is significantly reduced compared to naively sorting both the prefix and suffix for every possible f_t.

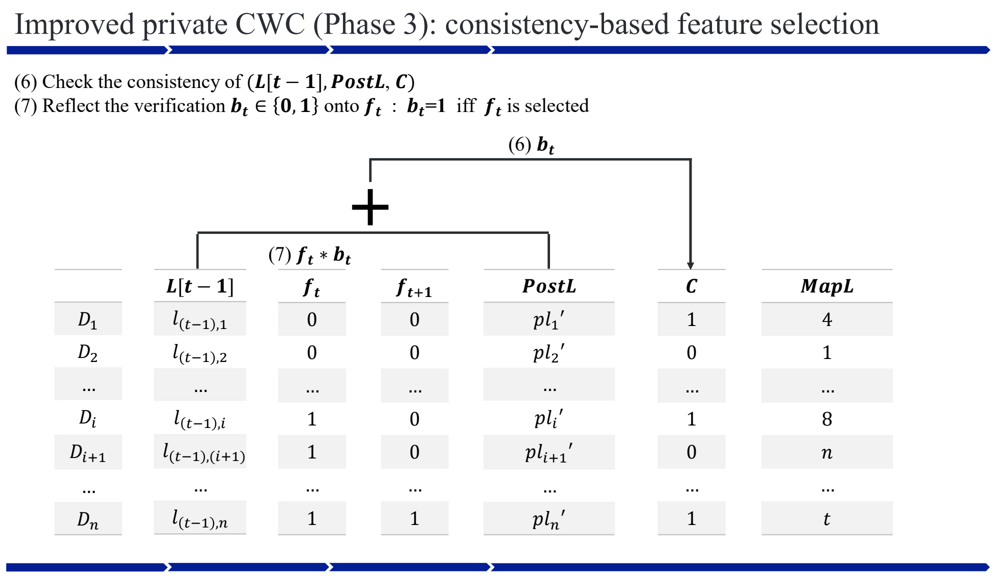
However, performing sorting on $\vec{F}[..t-1]$ breaks the alignment between the previously synchronized $\vec{F}[..t-1]$ and $\vec{F}[t+1..]$. To keep this correspondence, we introduce a reference array $MapL[1..n]$ initialized by $(1, 2, \ldots, n)$, and sort $(PostL[1..n], MapL[1..n])$ simultaneously. We then compute the inverse array $MapL^{-1}[1..n]$ by sorting $(1, 2, \ldots, n)$ with $MapL[1..n]$ as the key, and by sorting $MapL^{-1}[1..n]$ together with $\vec{F}[t+1..]$, the synchronization between the prefix and corresponding suffix of $\vec{F}$ is maintained. Through this procedure, we obtain the correctly sorted result of (D, F', C) without f_t excluded.

Figure 3 illustrates the task for verifying the consistency of the sorted (D, F', C) and determines whether f_t is a necessary feature. Since this information is extracted as an encrypted logical bit b_t, the decision can be reflected back to (D, F, C) by computing the product of b_t with all values of f_t. These operations are carried out sequentially in the order $t = k, k-1, \ldots, 1$, and by decrypting the resulting bit vector $\vec{b} = (b_1, b_2, \ldots, b_k)$, we obtain the final feature selection result.


Improved private CWC (Phase 2): feature labels for suffix $(f_t, f_{t-1}, \ldots, f_k)$
(3) Sort $(\boldsymbol{f_{t+1}}, \boldsymbol{PostL}, \boldsymbol{C}, \boldsymbol{MapL})$ with $\boldsymbol{f_{t+1}}$
(4) Update $\boldsymbol{PostL}$ by the sorted $\boldsymbol{f_{t+1}}$
(5) Sort $(\boldsymbol{f_t}, \boldsymbol{L}[\boldsymbol{t-1}], \boldsymbol{MapL^{-1}})$ with the inverted index $\boldsymbol{MapL^{-1}}$
$(\boldsymbol{L}[\boldsymbol{t-1}], \boldsymbol{PostL}, \boldsymbol{C})$ is synchronized based on the feature vector $(\boldsymbol{f_{t+1}}, \boldsymbol{f_{t+2}}, \ldots, \boldsymbol{f_k})$
(5)
(3)
(4)

	$L[t-1]$	f_t	f_{t+1}	$PostL$	C	$MapL$
D_1	$l_{(t-1),1}$	0	0	pl_1	1	4
D_2	$l_{(t-1),2}$	0	0	pl_2	0	1
…	…	…	…	…	…	…
D_i	$l_{(t-1),i}$	1	0	pl_i	1	8
D_{i+1}	$l_{(t-1),(i+1)}$	1	0	pl_{i+1}	0	n
…	…	…	…	…	…	…
D_n	$l_{(t-1),n}$	1	1	pl_n	1	i

Figure 2. Phase 2 (Line 4 -7) of Algorithm 3.


Improved private CWC (Phase 3): consistency-based feature selection
(6) Check the consistency of $(\boldsymbol{L}[\boldsymbol{t-1}], \boldsymbol{PostL}, \boldsymbol{C})$
(7) Reflect the verification $\boldsymbol{b_t} \in \{\boldsymbol{0}, \boldsymbol{1}\}$ onto $\boldsymbol{f_t}$: $\boldsymbol{b_t}$=1 iff $\boldsymbol{f_t}$ is selected
(6) $\boldsymbol{b_t}$
+
(7) $\boldsymbol{f_t} * \boldsymbol{b_t}$

	$L[t-1]$	f_t	f_{t+1}	$PostL$	C	$MapL$
D_1	$l_{(t-1),1}$	0	0	$pl_1{}'$	1	4
D_2	$l_{(t-1),2}$	0	0	$pl_2{}'$	0	1
…	…	…	…	…	…	…
D_i	$l_{(t-1),i}$	1	0	$pl_i{}'$	1	8
D_{i+1}	$l_{(t-1),(i+1)}$	1	0	$pl_{i+1}{}'$	0	n
…	…	…	…	…	…	…
D_n	$l_{(t-1),n}$	1	1	$pl_n{}'$	1	t

Figure 3. Phase 3 (Line 8 - 9) of Algorithm 3.

Theorem 2. *Algorithm 3 (improved pCWC) on ciphertexts simulates Algorithm 1 (CWC) on plaintexts. The time and space complexities are $O(kn \log^3 n)$ and $O(kn)$ for $|F| = k$ and $|D| = n$, respectively.*

Proof. We first show the correctness of Algorithm 3, that is it correctly simulates Algorithm 2. To establish this, it is sufficient to prove that for any feature f_t, Algorithm 3 (1) correctly sorts $(D, F \setminus \{f_t\}, C)$, and (2) accurately determines the consistency of the sorted $(D, F \setminus \{f_t\}, C)$.

As a result of preprocessing, for any t, (D, F, C) is already sorted using the prefix vector $\vec{F}[..t] = (f_1, f_2, \ldots, f_t)$ as the key, and thus the correct label $L[t][i]$ for D_i has been computed.

Now, assuming that the feature labels $PostL[1..n]$ corresponding to the suffix $\vec{F}[t+2..]$ has been computed in the previous loop, we can obtain the updated ranks for the extended suffix $\vec{F}[t+1..]$ by sorting $PostL[1..n]$ using f_{t+1} as the key. Then, by updating $PostL[1..n]$ based on the sorted values and their corresponding f_{t+1} values, we can compute the new $PostL[1..n]$ that reflects the extended suffix $\vec{F}[t+1..]$. All of these computations can be performed under FHE, in the same manner as in Algorithm 2.

The next necessary step is to restore the correspondence between $L[t-1][1..n]$ and $PostL[1..n]$. The array $L[t-1][1..n]$ has retained its original order from the initial sorting for prefixes, whereas the current $PostL[1..n]$ has been reordered according to the current suffix. Suppose that the previous $PostL[1..n]$ corresponding to the suffix $\vec{F}[t+2..]$ was already synchronized with $L[t-1][1..n]$.

To preserve the previous ordering of $PostL[1..n]$ (i.e., the ordering synchronized with $L[t-1][1..n]$), we first sort the initialized index array $MapL[1..n] = (1, 2, \ldots, n)$ together with $PostL[1..n]$. Then, since the original positions of $MapL[1..n]$ are retained in the inverse index array $MapL^{-1}[1..n]$, we can restore the correct correspondence between $L[t-1][1..n]$ and $PostL[1..n]$ by sorting $L[t-1][1..n]$ using $MapL^{-1}[1..n]$ as the key. All of these operations can be performed solely through sorting on encrypted data under FHE.

Moreover, since both f_{t+1} and C are always kept in synchronization with $PostL[1..n]$, and each f_t is always sorted in accordance with $L[t-1][1..n]$, Algorithm 3 correctly sorts $(D, F \setminus \{f_t\}, C)$. The consistency check applied to the sorted $(D, F \setminus \{f_t\}, C)$ is identical to that in Algorithm 2. Therefore, we conclude that the improved algorithm correctly simulates the naive one.

Next, we evaluate the computational complexity. The sorting of (D, F, C) in Lines 2–4 is identical to that of Algorithm 2, and thus requires $O(kn \log^3 n)$ time.

Among Lines 6–15, the most computationally expensive operations are the sorting steps in Line 8 and Line 10. In Line 8, sorting is performed using f_{t+1} as the key. To ensure stability, a $\log n$-bit suffix is appended to each entry, allowing the operation to be treated as sorting integers of $O(\log n)$ bits. Therefore, the time complexity is $O(n \log^3 n)$.

In Line 10, sorting is performed using $MapL[1..n]$ as the key. Since $MapL[1..n]$ is initialized as $(1, 2, \ldots, n)$, no additional suffix is required for stable sorting, and this operation also runs in $O(n \log^3 n)$ time. Because the above process is repeated k times, the overall computational time is $O(kn \log^3 n)$.

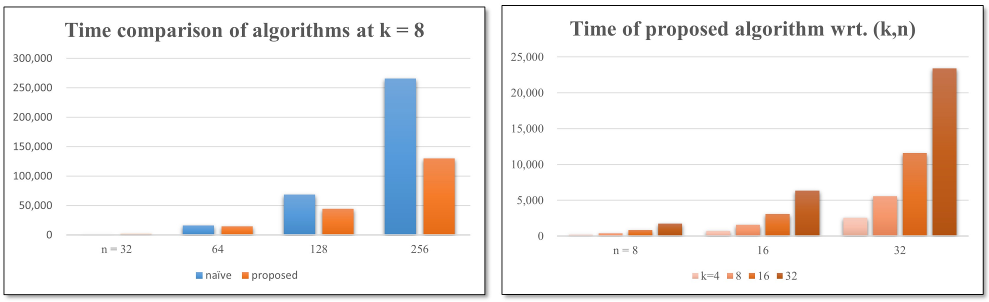
Finally, the additional data structures used by Algorithm 3, besides (D, F, C), include $L[1..k][1..n]$, $PostL[1..n]$, $MapL[1..n]$, and $MapL^{-1}[1..n]$. Since each of these has a size of $O(kn)$, the total space complexity of the algorithm is also $O(kn)$. □

5. Experimental Results

We implemented the proposed private CWC (Algorithm 3) in C++. For FHE operations, we used TFHE library [25]. The experiments were conducted on a machine equipped with an Intel(R) Core(TM) i9-10900X CPU running at 3.70GHz and 32 GB of memory. We used `gcc` version 11.4.0 as the compiler. All experiments were executed inside a Docker container specifically built for this purpose.

To evaluate the effectiveness of the proposed method (Algorithm 3), we compared it with the naive baseline approach (Algorithm 2). In the baseline algorithm, label computation becomes a bottleneck due to the need for k sorting operations, which makes the method impractical for large values of k. To address this issue, a decision-tree-based data structure can be used to classify encrypted feature vectors and assign labels without sorting [36,37]. Assuming this data structure, the computational complexity of the baseline algorithm becomes $O(kn^2 \log n)$, and then, the total complexity of Algorithm 2 can be considered as $O(\min\{k^2 n \log^3 n, kn^2 \log n\})$. In contrast, the improved method (Algorithm 3) has a complexity of $O(kn \log^3 n)$. We experimentally compared the execution times of these algorithms while varying k and n.

The experimental results are illustrated in Figure 4. Figure 4a compares the computation time of the proposed algorithm, which has a theoretical complexity of $O(kn \log^3 n)$, with a naive implementation that uses decision-tree-based secure computation [36–38] and has a complexity of $O(kn^2 \log n)$. In this experiment, we measured the computation time of both algorithms as the number of records n increases. The results clearly show that the proposed method reduces computation time even for relatively small values of n, and this advantage becomes more pronounced as n increases.



(**a**) (**b**)

Figure 4. Time (sec.) comparison of proposed and naive algorithms at several (k, n).

On the other hand, Figure 4b shows the impact of the parameters (k, n) on the computation time of the proposed algorithm. In this experiment, we compared the execution time for $k \in \{4, 8, 16, 32\}$ and $n \in \{8, 16, 32\}$. Although the theoretical time complexity of the proposed method is $O(kn \log^3 n)$, the results empirically confirm that the computation time increases linearly with k. Therefore, it is confirmed that the proposed privacy-preserving computation algorithm behaves as expected by design.

6. Discussion

In this study, we proposed a feature selection algorithm based on FHE. While various plaintext feature selection methods have been studied, consistency-based feature selection has been shown to offer both scalability and effectiveness. Therefore, the development of privacy-preserving computation protocols for consistency-based feature selection is a critical research challenge.

Although existing FHE-based two-party algorithms have addressed this goal to some extent, the protocol proposed in this work is the first to realize *fully outsourced* computation for consistency-based feature selection. We demonstrated that our proposed privacy-preserving algorithm is superior both theoretically and experimentally compared to a naive approach that uses decision-tree-based data structures under secure computation.

On the other hand, the proposed algorithm has several directions for future improvement. First, this work assumes that the input data (D, F, C) is binary. Removing this assumption and extending the algorithm to support multi-valued or symbolic attribute domains is one of the most important future enhancements. Since TFHE library [25] used in this study, is optimized for bitwise and integer operations, alternative libraries or methods capable of handling real-valued data more efficiently should be considered. In addition, while this study assumes a point-to-point computation model between data owner and analyst, extending the model to support one-to-many computations is also important for broader applicability. Such an extension could be realized by leveraging multi-key FHE [39–42].

By addressing these aspects, the proposed algorithm can be extended and adapted to enable privacy-preserving feature selection across a wide range of application domains such as FHE-based machine learning [43–47] and federated machine learning [48–51].

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

Funding: This research was funded in part by JSPS KAKENHI (Grant Number 23K11233, 23K28049, 21H05052), and JST AIP Acceleration Research JPMJCR24U4, Japan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable.

Conflicts of Interest: Not applicable.

References

1. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014** *40*, 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024
2. Pawlak, Z. Rough Sets, Theoretical aspects of reasoning about data. Kluwer Academic Publishers, 1991. https://doi.org/10.1007/978-94-011-3534-4
3. Liu, H.; Motoda, H.; Dash, M. A monotonic measure for optimal feature selection. In Proceedings of the 10th European Conference on Machine Learning, Chemnitz, Germany, 21–23 April 1998; pp. 101–106. https://doi.org/10.1007/bfb0026678
4. Zhao, Z.; Liu, H. Searching for interacting features. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007; pp. 1156–1161. http://ijcai.org/Proceedings/07/Papers/187.pdf
5. Dash, M.; Liu, H. Consistency-based search in feature selection. *Artificial Intell.* **2003**, *151*, 155–176. doi:10.1016/S0004-3702(03)00079-1
6. Arauzo-Azofra, A.; Benitez, J.M.; Castro, J.L. Consistency measures for feature selection. *J. Intell. Inf. Syst.* **2008**, *30*, 273–292. https://doi.org/10.1007/s10844-007-0037-0
7. Shin, K.; Fernandes, D.; Miyazaki, D. Consistency measures for feature selection: A formal definition, relative sensitivity comparison, and a fast algorithm. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Spain, 16–22 July 2011; pp.1491–1497. https://www.ijcai.org/Proceedings/11/Papers/251.pdf
8. Shin, K.; Kuboyama, T.; Hashimoto, T.; Shepard, D. SCWC/SLCC: Highly scalable feature selection algorithms. *Information* **2017**, *8*, 159. https://doi.org/10.3390/info8040159
9. Agarwal, A.; Boyle, E.; Chandran, N.; Gilboa, N.; Gupta, D.; Ishai, Y.; Kelkar, M.; Ma, Y. Secure sorting and selection via function secret sharing. *Proc. 2024 ACM SIGSAC Conf. Comput. Commun. Secur. (CCS'24)*, Salt Lake City, UT, USA, Oct 14–18, 2024; ACM: New York, NY, USA, 2024; pp 3023–3030. https://doi.org/10.1145/3658644.3690359
10. Dick, T.; Gillenwater, J.; Joseph, M. Better private linear regression through better private feature selection. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, USA, Dec 2023. https://dl.acm.org/doi/10.5555/3666122.3668448
11. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178. https://doi.org/10.1145/1536414.1536440
12. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91. https://doi.org/10.1007/s00145-019-09319-x
13. Chen, H.; Dai, W.; Kim, M.; Song, Y. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 395–412. https://doi.org/10.1145/3319535.33632
14. Liu, F.; Ng, W.K.; Zhang, W. Encrypted SVM for outsourced data mining. In Proceedings of the 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, USA, 20 August 2015; pp.1085–1092. https://doi.org/10.1109/CLOUD.2015.158

15. Qiu, G.; Huo, H.; Gui, X.; Dai, H. Privacy-preserving outsourcing scheme for SVM on vertically partitioned data. *Secur. Commun. Networks* **2022**, *2022*, 9983463. https://doi.org/10.1155/2022/9983463
16. Saeys, Y.; Inza, I.; Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics*, **2007**, *23*(19), 2507–2517. doi:10.1093/bioinformatics/btm344
17. Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V. Gene selection for cancer classification using support vector machines. *Machine Learning*, **2002**, *46*(1), 389–422. https://doi.org/10.1023/a:1012487302797
18. Breiman, L. Random forests. *Machine Learning*, **2001**, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324
19. Peng, H.; Long, F.; Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2005**, *27*(8), 1226–1238. https://doi.org/10.1109/tpami.2005.159
20. Liu, H.; Setiono, R. A probabilistic approach to feature selection – a filter solution. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, Morgan Kaufmann, Bari, Italy, 1996, pp. 319–327. https://api.semanticscholar.org/CorpusID:17123515
21. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, **1978**, *21*(2), 120–126. https://doi.org/10.1145/359340.359342
22. Goldwasser, S.; Micali, S. Probabilistic encryption. *J. Comput. Syst. Sci.* **1984**, *28*, 270–299.
23. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; pp. 223–238. https://doi.org/10.1016/0022-0000(84)90070-9
24. Boneh, D.; Goh, E.J.; Nissim, K. Evaluating 2-DNF formulas on ciphertexts. In Proceedings of the Theory of Cryptography Conference, Cambridge, MA, USA, 10–12 February 2005; pp. 325–341. https://doi.org/10.1007/978-3-540-30576-7_18
25. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryption library, August 2016. Available online: https://tfhe.github.io/tfhe (accessed on 28 January 2021).
26. Rao, V.; Long, Y.; Eldardiry, H.; Rane, S.; Rossi, R.A.; Torres, F. Secure two-party feature selection. *arXiv* **2019**. https://api.semanticscholar.org/CorpusID:57375741
27. Li, X.; Dowsley, R.; Cock, M.D. Privacy-preserving feature selection with secure multiparty computation. In Proceedings of the 38th International Conference on Machine Learning, Online, 18–24 July 2021; pp. 6326–6336. https://proceedings.mlr.press/v139/li21e/li21e.pdf
28. Akavia, A.; Galili, B.; Shaul, H.; Weiss, M.; Yakhini, Z. Privacy preserving feature selection for sparse linear regression. *IACR Cryptology ePrint Archive*, **2023**, *2023*, 1354. https://doi.org/10.56553/popets-2024-0017
29. Wang, L.; Guo, H.; Wu, W.; Zhou, L. Efficient and privacy-preserving feature selection based on multiparty computation. *IEEE Transactions on Information Forensics and Security*, **2025**, *20*, 3505–3518. http://dx.doi.org/10.1109/TIFS.2025.3546843
30. Ono, S.; Takata, J.; Kataoka, M.; I, T.; Shin, K.; Sakamoto, H. Privacy-preserving feature selection with fully homomorphic encryption. *Algorithms* **2022**, *15*, 229. https://doi.org/10.3390/a1010000
31. Garey, M.R.; Johnson, D.S. *Computers and intractability: a guide to the theory of NP-completeness*; Series of Books in the Mathematical Sciences, 1st ed.; W. H. Freeman and Company: New York, 1979; p.221–222. https://doi.org/10.1137/1024022
32. Bonnoron, G.; Fontaine, C.; Gogniat, G.; Herbert, V.; Lapôtre, V.; Migliore, V.; Roux-Langlois, A. Somewhat/Fully homomorphic encryption: implementation progresses and challenges. In *Codes, Cryptology and Information Security*; El Hajji, S., Nitaj, A., Souidi, E. M., Eds.; Springer International Publishing: Cham, 2017; pp 68–82. https://doi.org/10.1007/978-3-319-55589-8_5
33. Batcher, K.E. Sorting networks and their applications. In Proceedings of the American Federation of Information Processing Societies Spring Joint Computing Conference, Atlantic City, NJ, USA, 30 April–2 May 1968; pp. 307–314. https://doi.org/10.1145/1468075.1468121
34. Hamada, K.; Chida, K.; Ikarashi, D.; Takahashi, K. Oblivious radix sort: an efficient sorting algorithm for practical secure multi-party computation. *IACR Cryptol. ePrint Arch.* **2014** *121*. https://eprint.iacr.org/2014/121
35. Ajtai, M.; Szemerédi, E.; Komlós, J. An $O(n \log n)$ sorting network. In Proceedings of the 15th Annual ACM Symposium on Theory of Computing, Boston, MA, USA, 25–27 April 1983; pp.1–9. http://dx.doi.org/10.1145/800061.808726
36. Bost, R.; Popa, R.A.; Tu, S.; Goldwasser, S. Machine learning classification over encrypted data. In Proceedings of the 22nd Annual Network and Distributed System Security Symposium, San Diego, CA, USA, 2015. https://eprint.iacr.org/2014/331.pdf

37. Paul, J.; Tan, B.H.M.; Veeravalli, B.; Aung, K.M.M. Non-interactive decision trees and applications with multi-bit TFHE. *Algorithms* **2022**, *15*, 333. https://doi.org/10.3390/a15090333
38. Alabdulkarim, A.; Al-Rodhaan, M.; Ma, T.; Tian, Y. PPSDT. A novel privacy-preserving single decision tree algorithm for clinical decision-support systems using IoT devices. *Sensors* **2019**, *19*, 142. https://doi.org/10.3390/s19010142
39. Ma, J.; Naas, S.-A.; Sigg, S.; Lyu, X. Privacy-preserving federated learning based on multi-key homomorphic encryption. arXiv preprint **2021**, arXiv:2104.06824. https://doi.org/10.1002/int.22818
40. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Efficient multi-key homomorphic encryption with packed ciphertexts for secure neural network inference. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery: New York, NY, USA, 2019; pp. 395–412. https://doi.org/10.1145/3319535.3363207
41. Cheon, J.H.; Choe, H.; Kim, S.; Yeo, Y. Reusable dynamic multi-party homomorphic encryption. In Proceedings of the ACM CCS 2025, Seoul, Republic of Korea. **2025**. https://eprint.iacr.org/2025/581
42. Namazi, M.; Farahpoor, M.; Ayday, E.; Pérez-González, F. Privacy-preserving framework for genomic computations via multi-key homomorphic encryption. *Bioinformatics* **2025**, *41* (3), btae754. https://doi.org/10.1093/bioinformatics/btae754
43. Yuan, J.; Liu, W.; Shi, J.; Li, Q. Approximate homomorphic encryption based privacy-preserving machine learning: a survey. *Artif. Intell. Rev.* **2025**, *58*, 82. https://doi.org/10.1007/s10462-024-11076-8
44. Naresh, V.S.; Reddi, S. Exploring the future of privacy-preserving heart disease prediction: a fully homomorphic encryption-driven logistic regression approach. *J. Big Data* **2025**, *12*, 52. https://doi.org/10.1186/s40537-025-01098-6
45. Babu, K.M.; Syed, M.; Shaik, S.; Thalari, S.; Macha, U.; Chatakondu, A. Fully homomorphic encryption framework for privacy preserving in healthcare through decentralized machine learning. In *Challenges in Information, Communication and Computing Technology*, Sharmila, V., Ed.; CRC Press: London, UK, 2025; Volume 2, pp 812–819. https://doi.org/10.1201/9781003559092-140
46. Kolhar, M.; Aldossary, S.M. Privacy-preserving convolutional Bi-LSTM network for robust analysis of encrypted time-series medical images. *AI* **2023**, *4*, 706–720. https://doi.org/10.3390/ai4030037
47. Xiao, X.; Wu, T.; Chen, Y.; Fan, X. Privacy-preserved approximate classification based on homomorphic encryption. *Math. Comput. Appl.* **2019**, *24*, 92. https://doi.org/10.3390/mca24040092
48. Firdaus, M.; Larasati, H. T.; Hyune-Rhee, K. Blockchain-based federated learning with homomorphic encryption for privacy-preserving healthcare data sharing. *Internet Things* **2025**, *31*, 101579. https://doi.org/10.1016/j.iot.2025.101579
49. Zhang, C.; Zhang, X.; Yang, X.; Liu, B.; Zhang, Y.; Zhou, R. Poisoning attacks resilient privacy-preserving federated learning scheme based on lightweight homomorphic encryption. *Inf. Fusion* **2025**, *121*, 103131. https://doi.org/10.1016/j.inffus.2025.103131
50. Zhao, Y.; Liu, Y.; Tang, Q.; Peng, X.; Tan, X. Secure and flexible privacy-preserving federated learning based on multi-key fully homomorphic encryption. *Electronics* **2023**, *13* (22), 4478. https://doi.org/10.3390/electronics13224478
51. Walskaar, I.; Tran, M.C.; Catak, F.O. A practical implementation of medical privacy-preserving federated learning using multi-key homomorphic encryption and flower framework. *Cryptography* **2023**, *7* (4), 48. https://doi.org/10.3390/cryptography7040048