algorithms

MDPI

Article

Privacy-Preserving Feature Selection with Fully Homomorphic Encryption

Shinji Ono [1], Jun Takata [1], Masaharu Kataoka [1], Tomohiro I [1], Kilho Shin [2] and Hiroshi Sakamoto [1,*]

[1] Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi 820-8502, Japan; ono.shinji514@mail.kyutech.jp (S.O.); takata.jun903@mail.kyutech.jp (J.T.); kataoka.masaharu403@mail.kyutech.jp (M.K.); tomohiro@ai.kyutech.ac.jp (T.I.)

[2] Computer Centre, Gakushuin University, 1-5-1 Mejiro, Toshimaku, Tokyo 171-8588, Japan; kilhoshin314@gmail.com

* Correspondence: hiroshi@ai.kyutech.ac.jp

Abstract: For the feature selection problem, we propose an efficient privacy-preserving algorithm. Let D, F, and C be data, feature, and class sets, respectively, where the feature value $x(F_i)$ and the class label $x(C)$ are given for each $x \in D$ and $F_i \in F$. For a triple (D, F, C), the feature selection problem is to find a consistent and minimal subset $F' \subseteq F$, where 'consistent' means that, for any $x, y \in D$, $x(C) = y(C)$ if $x(F_i) = y(F_i)$ for $F_i \in F'$, and 'minimal' means that any proper subset of F' is no longer consistent. On distributed datasets, we consider feature selection as a privacy-preserving problem: assume that semi-honest parties A and B have their own personal D_A and D_B. The goal is to solve the feature selection problem for $D_A \cup D_B$ without sacrificing their privacy. In this paper, we propose a secure and efficient algorithm based on fully homomorphic encryption, and we implement our algorithm to show its effectiveness for various practical data. The proposed algorithm is the first one that can directly simulate the CWC (Combination of Weakest Components) algorithm on ciphertext, which is one of the best performers for the feature selection problem on the plaintext.

Keywords: CWC algorithm; oblivious sorting; TFHE; IND-CPA

Citation: 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/a15070229

Academic Editor: Quan Qian

Received: 30 May 2022
Accepted: 26 June 2022
Published: 30 June 2022

1. Introduction

1.1. Motivation

This study proposes a secure feature selection protocol that works effectively as a preprocessor for traditional machine learning (ML). Let us consider a scenario where different data owners are interested in private ML model training (e.g., logistic regression [1], SVM [2,3], and decision tree [4,5]) on their combined data. There is a large advantage to securely training these ML models on distributed data due to competitive advantage or privacy regulations. Feature selection is the problem of finding a subset of relevant features for model training. Using well-chosen features can lead to more accurate models, as well as speedup during model training [6].

Consider a data set D associated with a feature set F and a class variable C, where all feature values $x(F_i)$ $(F_i \in F)$ and the corresponding class label $x(C)$ are defined for each datum $x \in D$. In Table 1, for example, we show a concrete example. Given a triple (D, F, C), the feature selection problem is to find a minimal $F' \subseteq F$ that is relevant to the class C. The relevance of F' is evaluated, for example, by $I(F'; C)$, which measures the mutual information between F' and C. On the other hand, F' is minimal, if any proper subset of F' is no longer consistent.

To the best of our knowledge, the most common method for identifying favorable features is to choose features that show higher relevance in some statistical measures. Individual feature relevance can be estimated using statistical measures such as mutual information and Bayesian risk. For example, at the bottom row of Table 1, the mutual information score $I(F_1; C)$ of each feature F_i to class labels is described. We can see that F_1

is more important than F_5, because $I(F_1;C) > I(F_5;C)$. F_1 and F_2 of Table 1 will be chosen to explain C based on the mutual information score. However, a closer examination of D reveals that F_1 and F_2 cannot uniquely determine C. In fact, we find x_2 and x_5 with $x_2(F_1) = x_5(F_1)$ and $x_2(F_2) = x_5(F_2)$ but $x_2(C) \neq x_5(C)$. 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 $I(F_4;C) = I(F_5;C) = 0$. As a result, the traditional method based on individual feature relevance scores misses the correct answer.

Table 1. An example dataset shown in [7].

D	F_1	F_2	F_3	F_4	F_5	C
x_1	1	0	1	1	1	0
x_2	1	1	0	0	0	0
x_3	0	0	0	1	1	0
x_4	1	0	1	0	0	0
x_5	1	1	1	1	0	1
x_6	0	1	0	1	0	1
x_7	0	1	0	0	1	1
x_8	0	0	0	0	1	1
$I(F_i;C)$	0.189	0.189	0.049	0.000	0.000	

Thus, we concentrate on the concept of consistency: $F' \subseteq F$ is considered to be consistent if, for any $x, y \in D$, $x(F_i) = y(F_i)$ for all $F_i \in F'$ implies $x(C) = y(C)$. In machine learning research, consistency-based feature selection has received a lot of attention [8–12]. CWC (Combination of Weakest Components) [8] is the simplest of such consistency-based feature selection algorithms, and even though CWC uses the most rigorous measure, it shows one of the best performances in terms of accuracy as well as computational speed compared to other methods [7]. Throughout the proposed secure protocol, none of the parties learns the values of the data as all computations are done over ciphertexts. Next, the parties train an ML model over the pre-processed data using existing privacy-preserving training protocols (e.g., logistic regression training [13] and decision tree [14]). Finally, they can disclose the trained model for common use.

To design a secure protocol for feature selection, we focus on the framework of homomorphic encryption. Given a public key encryption scheme E, let $E[m]$ denote a ciphertext of integer m; if $E[m+n]$ can be computed from $E[m]$ and $E[n]$ without decrypting them, then E is said to be *additive* homomorphic, and if $E[mn]$ can also be computed, then E is said to be *fully* homomorphic. Furthermore, modern public key encryption must be *probabilistic*: when the same message m is encrypted multiple times, the encryption algorithm produces different ciphertexts of $E[m]$.

Various homomorphic encryption schemes have been proposed to satisfy these homomorphic properties over the last two decades. The first additive homomorphic encryption was proposed by Paillier [15]. *Somewhat* homomorphic encryption that allows a sufficient number of additions and a limited number of multiplications has also been proposed [16–18], and we can use these cryptosystems to compute more difficult problems, such as the inner product of two vectors. Gentry [19] proposed the first fully homomorphic encryption (FHE) with an unlimited number of additions and multiplications, and since then, useful libraries for fully homomorphic encryption have been developed, particularly for bitwise operations and floating-point operations. TFHE [20,21] is known as the fastest fully homomorphic encryption that is optimized for bitwise operations.

For the private feature selection problem, we use TFHE to design and implement our algorithm. In this case, we assume two *semi-honest* parties A and B: each party complies with the protocol but tries to infer as much as possible about the secret from the information obtained. The parties have their own private data D_{A} and D_{B} and they jointly compute advantageous features for $D_{\mathsf{A}} \cup D_{\mathsf{B}}$ while maintaining their privacy. The goal is to jointly compute the CWC algorithm result on $D = D_{\mathsf{A}} \cup D_{\mathsf{B}}$ without revealing any other information.

In this paper, we describe the simplest case where there are two data owners, and they perform the cooperative secure computation. More generally, there are many data owners, and they encrypt with their own public keys. Since homomorphic operations cannot be applied to two data encrypted with different public keys, a simple approach would be for the server to attempt to re-encrypt them with some common public key. However, there is no guarantee that the server or the new public key can be trusted. To solve this problem, the framework of multi-key homomorphic encryption was proposed. This allows FHE operations on data encrypted with different keys, i.e., we can extend the two-party computation model to a more general case because TFHE has the required property. Using this property, its application to the framework of oblivious neural network inference [22] has been proposed.

This should be a realistic requirement, if one wants to draw some conclusions from data that are privately distributed over more than one party. Multi-party computation (MPC) can provide effective technical solutions to realize this requirement in many cases. In MPC, certain computations that essentially rely on the distributed data are performed through cooperation among the parties. In particular, fully homomorphic encryption (FHE) is one of the critical tools of MPC. One of the most significant advantages of FHE-based MPC is thought to be that FHE realizes outsourced computation in a simple and straightforward manner: parties encrypt their private data with their public keys and send the encrypted data to a single trusted party with sufficient computational power to perform the required computation; although the computational results of the trusted party may be incorrect, if some malicious parties send incorrect data, honest parties are at least convinced that their private data have not been stolen as far as the cryptosystem used is secure. In contrast, when a party shares his/her secret with other parties to perform MPC, even if it uses a secure secret sharing scheme, collusion of a sufficient number of compromised parties may reveal the party's secret. In general, it is difficult to prove the security of MPC protocols for the situation where we cannot deny the existence of active malicious parties, and hence, the security is very often proven assuming that all the parties are at worst semi-honest. In reality, however, even this relaxed assumption is unable to hold. Thus, the property that a party can protect its private data only relying on its own efforts should be counted as an important advantage of FHE-based MPC.

On the other hand, the current implementations of FHE are thought to be significantly inefficient, and consequently, their ranges of application are actually limited. This is currently true, but may not be true in the future: the Goldwasser–Mmicali (GM) cryptosystem [23] is considered as the first scheme with provable security. Unfortunately, because the GM cryptosystem encrypts data in a bitwise manner, it has turned out not to have sufficient efficiency in time and memory to be used in the real world. In 2001, however, RSA-OAEP was finally proven to have both provable security and realistic efficiency [24,25], and is widely used through SSL/TLS. Thus, studying FHE-based MPC does not merely have theoretical meaning, but also will yield significant contributions in terms of application to the real world in the future.

In this paper, we propose an MPC protocol which relies on FHE-based outsourced computation as well as mutual cooperation among parties. The target of our protocol is to perform the computation of CWC, a feature selection algorithm known to be accurate and efficient, preserving the privacy of the participating parties. If we fully perform CWC by FHE-based outsourced computation, we have to pay unnecessarily large costs in time in the phase of sorting the features of CWC. Therefore, in our proposed scheme, we add ingenuity so that two parties cooperate with each other to sort the features efficiently.

Converting CWC into its privacy-preserving version based on different primitives of MPC—for example, based on secret sharing techniques—is not only interesting but also useful both in theory and in practice. We will pursue this direction as well in our future work.

1.2. Our Contribution and Related Work

Table 2 summarizes the complexities of the proposed algorithms in comparison to the original CWC on plaintext. The *baseline* is a naive algorithm that can simulate the original CWC [8] over ciphertext using TFHE operations. The bottleneck of private feature selection exists in the sorting task over ciphertext, as we mention in the related work below. Our main contribution is the improved algorithm, shown as 'improved', which significantly reduces the time complexity caused by the sorting task. We also implement the improved algorithm and demonstrate its efficiency through experiments in comparison to the baseline.

Table 2. Time and space complexities of the baseline and improved algorithms for secure CWC, where k is the number of features and m, n are the numbers of positive and negative data, respectively. We assume that the time of the respective operation (e.g., encryption/addition/multiplication/comparison) in FHE is $O(1)$.

Algorithm	Time	Space
CWC on plaintext [8]	$O(kmn + k \log k)$	$O(kmn)$
secure CWC (baseline)	$O(kmn \log k + k \log^2 k)$	$O(kmn)$
improved	$O(kmn + k \log^2 k + k \log k \log mn)$	$O(kmn)$

There are mainly two private computation models, secret sharing-based MPC and public key-based MPC, and secret sharing-based MPC currently has an advantage. On the other hand, we focus on the convenience of FHE. Public key-based MPC can establish a simple mechanism to obtain results while keeping the learning model possessed by the server and the personal information of many data owners confidential from each other, relying only on cryptographic strength. The secret sharing-based MPC is faster but requires at least two trusted parties that do not collude with each other, which creates a different problem to cryptographic strength.

Other drawbacks of public key-based MPC are its security against the chosen plaintext attack (CPA) and computational cost. TFHE is, however, computationally secure against the chosen ciphertext attack (CCA), which assumes a stronger adversary than CPA so that an attacker cannot obtain meaningful information from plaintext or ciphertext within polynomial time.

In this section, we discuss related work on private feature selection as well as the benefits of our method. Rao [26] et al. proposed a homomorphic encryption-based private feature selection algorithm. Their protocol allows the additive homomorphic property only, which invariably leaks statistical information about the data. Anaraki and Samet [27] proposed a different method based on the rough set theory, but their method suffers from the same limitations as Rao et al., and neither method has been implemented. Banerjee et al. [28], and Sheikhalishahi and Martinellil [29] have proposed MPC-based algorithms that guarantee security by decomposing the plaintext into shares, as a different approach to the private feature selection, while achieving cooperative computation. Li et al. [30] improved the MPC protocol on the aforementioned flaw and demonstrated its effectiveness through experiments.

These methods avoid partial decoding under the assumption that the mean of feature values provides a good criterion for feature selection. This assumption, however, is heavily dependent on data. The most important task in general feature selection is feature value-based sorting, and CWC and its variants [7,8,11] demonstrated the effectiveness of sorting with the consistency measure and its superiority over other methods. On ciphertext, this study realizes the sorting-based feature selection algorithm (e.g., CWC).

We focus on the learning decision tree by MPC [31] as another study that employs sorting for private ML, where the sorting is limited to the comparison of N values of fixed length in $O(N \log^2 N)$ time by a sorting network. In the case of CWC, however, the algorithm must sort N data points, each of which has a variable length of up to M, so a naive method requires $O(MN \log N + N \log^2 N)$ time. Our algorithm reduces this

complexity to $O(MN + N\log^2 N + N\log N\log M)$, which is significantly smaller than the naive algorithm depending on M and N. Through experiments, we confirm this for various data, including real datasets for ML.

Although sorting itself is not ML, a fast-sorting algorithm is an important preprocess for ML model training. In the previous result [7], it was shown that sorting-based feature selection can classify with higher accuracy than other heuristic methods. Furthermore, preprocessing by sorting has proven to be an important task in decision tree model training [31]. On the other hand, it is also well known that sorting can speed up ML model training. For example, in SVM, which is widely used in text classification and pattern recognition, the problem of finding the convex hull of n points in Euclidean space can be reduced from $O(n^2)$ to $O(nlogn)$ time by preprocessing it with an appropriate sorting algorithm.

2. Preliminaries

2.1. Consistency Measure

First of all, we review the notion of the consistency measure employed in our problem. A consistency measure $\mu : 2^F \to [0, \infty)$ for a feature set F is a function to represent how far the data deviate from a consistent state and is required to satisfy *determinisity* ($\mu(F) = 0$ if and only if F is consistent) and *monotonicity* ($F \subseteq G$ implies $\mu(F) \geq \mu(G)$). The following consistency measures satisfy this requirement.

- $\mu_{\text{bin}}(F) = 0$, F is consistent; 1, otherwise (binary consistency [11])
- $\mu_{\text{icr}}(F) = \sum_x (\Pr(F = x) - \max_c \Pr(F = x, C = c))$ (ICR [32])
- $\mu_{\text{rs}}(F) = 1 - \sum_c \frac{FD_{C=c}}{|D|}$, $FD = \{D_{F=x} \mid D_{F=x} \subseteq D\}$ (rough set [33])
- $\mu_{\text{ie}}(F) = \sum_x \sum_{c \neq c'} \frac{|D_{F=x,C=c}| \cdot |D_{F=x,C=c'}|}{|D|^2}$ (inconsistent pair [34])

The fully homomorphic encryption used in this study is specialized for binary operations. Therefore, among these consistency measures, we employ μ_{bin}.

2.2. CWC Algorithm over Plaintext

We generally assume that the dataset D associated with F and C contains no errors, i.e., if $x(F_i) = y(F_i)$ for all i, $x(C) = y(C)$. When D contains such errors, they are removed beforehand and D contains not more than one $x \in D$ with the same feature values.

In Algorithm 1, we describe the original algorithm for finding a minimal consistent feature for two-class data. Given D with F_i and $C = \{0, 1\}$, a datum $x \in D$ of $x(C) = 1$ is referred to as a positive datum and $y \in D$ of $y(C) = 0$ is referred to as a negative datum. Let n represent the number of positive data and $m = |D| - n$. We consider two-dimensional bit array $\mathbf{B}_i[1..n][1..m]$ such that, for any $1 \leq p \leq n$ and $1 \leq q \leq m$, $\mathbf{B}_i[p][q] = 0$ if $x_p(F_i) = y_q(F_i)$ and $\mathbf{B}_i[p][q] = 1$ otherwise, where x_p is the p-th positive datum ($1 \leq p \leq n$) and y_q is the q-th negative datum ($1 \leq q \leq m$). $\mathbf{B}_i[p][q] = 0$ means that F_i is not consistent with the pair (x_p, y_q) because $x_p(F_i) = y_q(F_i)$ despite $x_p(C) \neq y_q(C)$. Recall that F_i is said to be consistent only if $x(F_i) = y(F_i)$ implies $x(C) = y(C)$ for any $x, y \in D$. As a result, $||\mathbf{B}_i||$ is defined to be the number of 1s in $\mathbf{B}_i$.

For a subset $F' \subseteq F$, F' is said to be consistent, if for any $p \in [1, n]$ and $q \in [1, m]$, there exists i such that $F_i \in F'$ and $\mathbf{B}_i[p][q] = 1$ hold. CWC uses this to remove irrelevant features from F in order to build a minimal consistent feature set. We note that finding the smallest consistent feature set is clearly NP-hard. There is a simple reduction from the minimum set cover to this problem as follows: given $S_1, \ldots, S_k \subseteq S$ ($|S| = n$) with the intention that S_i is regarded as $\mathbf{B}_i$ in CWC, covering any element of S corresponds to the condition that for any $j \in \{1, \ldots, n\}$, there exists at least one i such that $\mathbf{B}_i[j] = 1$.

Since the point of $\mathbf{B}_i$ is that it contains information, for every pair of data across different classes, whether F_i is consistent with the pair or not, it can be easily extended to multi-class data that have more than two classes. Although we focus on two-class data for the sake of simplicity, for multi-class data, the mn-factors in the complexities are replaced with the number of pairs of data across different classes, which is upper bounded by $|D|(|D| - 1)/2$. Moreover, in extending to multi-class data, it is convenient to consider

$\mathbf{B}_i$ as an appropriately serialized one-dimensional bit string because there is no way to represent it as a dense two-dimensional bit array. Hence, in what follows, we treat $\mathbf{B}_i$ as a bit string.

Algorithm 1 The algorithm CWC for plaintext

```
Input: A dataset D associated with features F = {F_1, ..., F_k} and class C = {0,1}.
Output: A minimal consistent subset S ⊆ F.
Sort F_1, ..., F_k in the incremental order of ||B_i||.
Let π be the sorted indices of {1, ..., k}.
for i = 1, ..., k do
    if F \ {F_π[i]} is consistent then
        update F ← F \ {F_π[i]}
    end if
end for
```

Table 3 shows an example of D, and Table 4 shows the corresponding $\mathbf{B}_i$. Consider the behavior of CWC in this case. All $\mathbf{B}_i$ $(1 \leq i \leq 4)$ are computed as preprocessing. Then, the features are sorted by the order $||\mathbf{B}_2|| = 5 \leq ||\mathbf{B}_4|| = 5 \leq ||\mathbf{B}_3|| = 6 \leq ||\mathbf{B}_1|| = 8$ and $\pi = (2,4,3,1)$. By the consistency order π, CWC checks whether $F_{\pi[i]}$ can be removed from the current F. Using the consistency measure, CWC removes F_2 and F_4 and the resulting $\{F_1, F_3\}$ is the output. In fact, we can predict the class of x by the logical operation $\overline{x(F_1)} \wedge x(F_3)$.

Table 3. An example dataset D with $F = \{F_1, F_2, F_3, F_4\}$ and $C = \{0,1\}$. The data consist of two positive data $\{x_1, x_2\}$ and five negative data $\{y_1, y_2, y_3, y_4, y_5\}$.

$x_i \in D$	F_1	F_2	F_3	F_4	C
x_1	0	1	1	0	1
x_2	0	0	1	1	1
$y_i \in D$	F_1	F_2	F_3	F_4	C
y_1	1	0	1	0	0
y_2	1	1	0	0	0
y_3	0	1	0	1	0
y_4	1	0	1	0	0
y_5	1	1	0	0	0

Table 4. The bit string $\mathbf{B}_i$ for the example dataset D of Table 3. Each column (x_p, y_q) is 0 if $x_p(F_i) = y_q(F_i)$. For example, $\mathbf{B}_1 = (1,1,0,1,1,1,1,0,1,1)$ because $x_p(F_1) = y_q(F_1)$ only for the two pairs (x_1, y_3) and (x_2, y_3).

$\mathbf{B}_i$	(x_1, y_1)	(x_1, y_2)	(x_1, y_3)	(x_1, y_4)	(x_1, y_5)	(x_2, y_1)	(x_2, y_2)	(x_2, y_3)	(x_2, y_4)	(x_2, y_5)
$\mathbf{B}_1$	1	1	0	1	1	1	1	0	1	1
$\mathbf{B}_2$	1	0	0	1	0	0	1	1	0	1
$\mathbf{B}_3$	0	1	1	0	1	0	1	1	0	1
$\mathbf{B}_4$	0	0	1	0	0	1	1	0	1	1

2.3. Security Model

2.3.1. Indistinguishable Random Variables

Let $\mathbb{N}$ denote the set of natural numbers. A function $\epsilon : \mathbb{N} \to [0,1]$ is called *negligible*, if $\forall c > 0,\ \exists k,\ \forall n \geq k,\ \epsilon(n) < 1/n^c$. Let $X = \{X_k \mid k \in \mathbb{N}\}$ and $Y = \{Y_k \mid k \in \mathbb{N}\}$ be sequences of random variables such that X_k and Y_k are defined over the same sample space. We say that X and Y are *indistinguishable*, denoted by $X \equiv_c Y$, if, and only if, $\Pr[X_n = Y_n]$ is a negligible function.

2.3.2. Security of Multi-Party Computation (MPC)

Although the discussion of this section can be extended to MPC schemes which involve more than two parties, merely for simplicity, we focus on the case where only two parties are involved.

A two-party protocol is a pair $\Pi = (\mathcal{P}_1, \mathcal{P}_2)$ of PPT Turing machines with input and random tapes. Let x_i be an input of $\mathcal{P}_i$ and y_i be an output of $\mathcal{P}_i$, respectively.

We assume a *semi-honest adversary* $\mathcal{A}$ and consider a protocol $(\mathcal{A}, \mathcal{P}_2)$, replacing $\mathcal{P}_1$ in Π by $\mathcal{A}$, where $\mathcal{A}$ takes x_1 as input and apparently follows the protocol. Let $\mathtt{REAL}_{\Pi,\mathcal{A}}(x_1, x_2)$ denote the random variable representing the output (y_1, y_2) of $(\mathcal{A}, \mathcal{P}_2)$, and we define the class $\mathtt{REAL}_{\Pi,\mathcal{A}} = \{\mathtt{REAL}_{\Pi,\mathcal{A}}(x_1, x_2)\}_{x_1,x_2} = \{y_1, y_2\}_{x_1,x_2}$.

On the other hand, let $\mathcal{F}$ denote the functionality that the protocol Π is trying to realize, i.e., $\mathcal{F}$ is a PPT that simulates the honest $(\mathcal{P}_1, \mathcal{P}_2)$ so that $\mathcal{F}(x_1, x_2) \equiv (\mathcal{P}_1(x_1), \mathcal{P}_2(x_2))$. Here, we assume a completely reliable third party, denoted by $\mathcal{F}$. In this *ideal* world, for this $\mathcal{F}$ and any adversary $\mathcal{B}$ acting as $\mathcal{P}_1$ with input x_1', possibly $x_1' \neq x_1$, we define the random variable $\mathtt{IDEAL}_{\mathcal{F},\mathcal{B}}(x_1, x_2) = (\mathcal{B}(x_1, \mathcal{F}_1(x_1', x_2), \mathcal{F}_2(x_1', x_2)))$, where $\mathcal{F}_i(\cdot,\cdot)$ denotes the i-th component of the output of $\mathcal{F}(\cdot,\cdot)$ for $i = 1, 2$. Similarly, we denote the class $\mathtt{IDEAL}_{\mathcal{F},\mathcal{B}} = \{\mathtt{IDEAL}_{\mathcal{F},\mathcal{B}}(x_1, x_2)\}_{x_1,x_2}$.

Using such random variables, we define the security of protocol Π as follows.

Definition 1. *It is said that a protocol* Π *securely realizes a functionality* $\mathcal{F}$ *if, for any attacker* $\mathcal{A}$ *against* Π*, there exists an adversary* $\mathcal{B}$*, and* $\mathtt{REAL}_{\Pi,\mathcal{A}} \equiv_c \mathtt{IDEAL}_{\mathcal{F},\mathcal{B}}$ *holds.*

The definitions stated above can be intuitively explained as follows. Exactly conforming to the protocol, a semi-honest adversary $\mathcal{A}$ plays the role of $\mathcal{P}_1$ to steal any secrets. The information sources which $\mathcal{A}$ can take advantage of are the following three:

1. the input tape to $\mathcal{P}_1$;
2. the conversation with $\mathcal{P}_2$;
3. the execution of the protocol.

While the information that $\mathcal{A}$ can obtain from the first and third sources is exactly x_1 and y_1, respectively, we call the information from the second source a *view*.

To denote it, we use the symbol $\mathsf{View}_{\mathcal{P}_1}$.

Since the protocol inevitably requires that $\mathcal{A}$ obtains the information of x_1 and y_1, the security of the protocol questions what $\mathcal{A}$ can obtain, in addition to what can be computationally inferred from x_1 and y_1. If there exists such information, its source must be $\mathsf{View}_{\mathcal{P}_1}$.

The security criterion of *simulatability* requires that View_{P_1} can be simulated on the input of x_1 and y_1. In more formal terms, there exists a PPT Turing machine Sim that outputs a view on the input of x_1 and y_1 such that the output view cannot be distinguished from $\mathsf{View}_{\mathcal{P}_1}$ by any PPT Turing machine. When $\mathsf{View}_{\mathcal{P}_1}$ is simulatable, we see that Sim can generate by itself what Sim can obtain from View_{P_1}. Therefore, Sim cannot cannot obtain any information in addition to what Sim can compute from x_1 and y_1.

2.3.3. IND-CPA

Indistinguishability against chosen plaintext attack (IND-CPA) is an important criterion for the secrecy of a public key cryptosystem. We let $\Pi = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ denote a public key cryptosystem consisting of key generation, encryption, and decryption algorithms. To describe IND-CPA, we introduce the *IND-CPA game* played between an adversary $\mathcal{A}$ and an oracle $\mathcal{O}$: $\mathcal{A}$ is a PPT Turing machine, and k is the security parameter.

1. $\mathcal{O}$ generates a public key pair $(sk, pk) \leftarrow \mathtt{Gen}(1^k)$.
2. $\mathcal{A}$ generates two messages (m_0, m_1) of the same length arbitrarily and throws a query (m_0, m_1) to $\mathcal{O}$.
3. On receipt of (m_0, m_1), $\mathcal{O}$ selects $b \in \{0, 1\}$ uniformly at random, computes $c = \mathtt{Enc}(pk, m_b)$, and replies to $\mathcal{A}$ with c.

4. $\mathcal{A}$ guesses on b by examining c and outputs the guess bit b'.

We view b and b' as random variables whose underlying probability space is defined to represent the choices of the public key pair, b and b'. The *advantage* of the adversary $\mathcal{A}$ is defined as follows to represent the advantage of $\mathcal{A}$ over tossing a fair coin to guess $\mathcal{O}$'s secret b:

$$\mathsf{Adv}_{\mathcal{A}} = 2 \cdot \Pr[b' = b] - 1.$$

when we let

$$\Pr[b' = 0 | b = 0] = \frac{1}{2} + \alpha_0 \text{ and } \Pr[b' = 1 | b = 1] = \frac{1}{2} + \alpha_1,$$

We have

$$\mathsf{Adv}_{\mathcal{A}} = \alpha_0 + \alpha_1$$

This definition of the advantage is consistent with the common definition found in many textbooks:

$$\mathsf{Adv}_{\mathcal{A}} = \Pr[b' = 0 | b = 0] - \Pr[b' = 1 | b = 0]$$

Definition 2. *A public key cryptosystem* Π *is secure in the sense of IND-CPA, or simply IND-CPA secure, if* $\mathsf{Adv}_{\mathcal{A}}$ *as a function in* k *is a negligible function.*

2.4. TFHE: A Faster Fully Homomorphic Encryption

The proposed private feature selection is based on FHE. We review the TFHE [21], one of the fastest libraries for bitwise addition (this means XOR '$\oplus$') and bitwise multiplication (AND '$\cdot$') over ciphertext. On TFHE, any integer is encrypted bitwise: For ℓ-bit integer $m = (m_1, \ldots, m_\ell)$, we denote its bitwise encryption by $E[m] \equiv (E[m_1], \ldots, E[m_\ell])$, for short. These bitwise operations are denoted by $f_\oplus(E[x], E[y]) \equiv E[x \oplus y]$ and $f_\cdot(E[x], E[y]) \equiv E[x \cdot y]$ for $x, y \in \{0, 1\}$ and the ciphertexts $E[x]$ and $E[y]$. The same symbol is used to represent an encrypted array. For example, when x and y are integers of length ℓ and ℓ', respectively, $E(x, y)$ denotes

$$E[x, y] \equiv (E[x], E[y]) \equiv ((E[x_1], \ldots, E[x_\ell]), (E[y_1], \ldots, E[y_{\ell'}])).$$

TFHE allows all arithmetic and logical operations via the elementary operations $E[x \oplus y]$ and $E[x \cdot y]$. In this section, we will consider how to build the adder and comparison operations. Let x, y represent ℓ-bit integers and x_i, y_i represent the i-th bit of x, y, respectively. Let c_i represent the i-th carry-in bit and s_i is the i-th bit of the sum $x + y$. Then, we can obtain $E[x + y]$ by the bitwise operations of ciphertexts using $s_i = x_i \oplus y_i \oplus c_i$ 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 adder. For example, $E[x - y]$ is obtained by $E[x + (-y)]$, where $(-y)$ is the bit complement of y obtained by $y_i \oplus 1$ for all i-th bits. On the other hand, we examine the comparison. We want to obtain $E[x <?y]$ without decrypting x and y where $x <?y = 1$ if $x < y$ and $x <?y = 0$ otherwise. We can obtain the logical bit for $x <?y$ as the most significant bit of $x + (-y)$ over ciphertexts here. Similarly, for the equality test, we can compute the encrypted bit $E[x =?y]$.

Adopting these operations of TFHE, we design a secure multi-party CWC. In this paper, we omit the details of TFHE (see, e.g., [20,21]).

We should note that the secrecy of TFHE definitely impacts the security of our scheme. In fact, in our two-party feature selection scheme, the party B sends his/her inputs in an encrypted form to the party A, and A performs the computation of feature selection on the encrypted inputs. If the encrypted inputs could be easily infiltrated, any ingenious devices to secure the scheme would be meaningless.

Therefore, in designing our scheme, it was a matter of course to require our FHE cryptosystem to be IND-CPA secure. In fact, TFHE is known to be IND-CPA secure. Regarding this, we should note the following:

- By definition, encryption with an ID-CPA cryptosystem is probabilistic. That is, the result $E[x]$ of encryption unpredictably differs every time the encryption is performed. For this reason, by $E[x|t]$, we denote a ciphertext generated at time t. In particular, the notation of $E[x|*]$ means that the ciphertext has been generated at a time different from any other encryption events.
- When we consider the IND-CPA security of an FHE cryptosystem, we should note that the way in which the oracle $\mathcal{O}$ generates c with $D(c) = m_b$ is not unique. For example, the oracle may compute c from two ciphertexts of additive shares of m_b, say $E[r]$ and $E[m_b \oplus r]$, by $c = f_{\oplus}(E[r], E[m_b \oplus r])$. The IND-CPA security of an FHE cryptosystem should require that $\mathcal{A}$ cannot guess b with effective advantage, no matter how c has been generated. This, however, holds, if the result of performing $E[x], f_{\oplus}(E[x], E[y])$ and $f_{\cdot}(E[x], E[y])$ distributes uniformly, and TFHE is known to satisfy this condition.

3. Algorithms

3.1. Baseline Algorithm

We present the baseline algorithm, a privacy-preserving variant of CWC. In this subsection, we consider a two-party protocol, in which a party B has its private data and outsources CWC computation to another party A, where this protocol can be extended to more than two data owners using the multi-key homomorphic encryption [22]. During the computation, party A should not gain other information than the number n of positive data, the number m of negative data, and the number k of features. It should be noted that party B can hide the actual number of data by inserting dummy data and telling A the inflated numbers n and m. Dummy data can be distinguished by adding an extra bit that indicates that the datum is a dummy if the bit is 1. The values of features and dummy bits of data in each class are encrypted by B's public key and sent to A.

The baseline algorithm consists of three tasks: computing encrypted bit string $E[\mathbf{B}_i]$, sorting $E[\mathbf{B}_i]$s, and executing feature selection on $E[\mathbf{B}_i]$s. In the baseline algorithm, all inputs are encrypted and they are not decrypted until the computation is completed. Thus, for simplicity, we omit the notation E in the following presentation.

3.1.1. Computing $\mathbf{B}_i$

We can compute $\mathbf{B}_i[m(p-1)+q]$ by $(x_p(F_i) \oplus y_q(F_i)) \vee x_p(d) \vee y_q(d)$, where $x_p(d)$ and $y_q(d)$ represent the dummy bits for data x_p and y_q, respectively. $(x_p(F_i) \oplus y_q(F_i))$ becomes 0 if F_i is inconsistent for the pair of x_p and y_q. Since we want to ignore the influence of dummy data, the part '$\vee x_p(d) \vee y_q(d)$' is added to make the whole value 1 (meaning that it is consistent) when one of x_p and y_q is a dummy. It takes $O(kmn)$ time and space in total.

3.1.2. Sorting **B**s

We can compute $\|\mathbf{B}_i\|$ in encrypted form by summing up the values in $\mathbf{B}_i$ in $O(mn \log(mn))$ time (noting that each operation on integers of $\log(mn)$ bits takes $O(\log(mn))$ time). Instead, we can set an upper bound b_{max} of the bits used to store the consistency measure to reduce the time complexity to $O(mnb_{max})$.

Then, sorting **B**s in the incremental order of consistency measures can be accomplished using any sorting network in which comparison and swap are performed in encrypted form, without leaking information about feature ordering. It should be noted that in this approach, the algorithm must spend $\Theta(mn + \log k)$ time to swap (or pretend to swap) two-bit strings and original feature indices of $\log k$ bits regardless of whether the two features are actually swapped or not. Because this is the most complex part of our baseline algorithm, we will demonstrate how to improve it. Using an AKS sorting network [35] of size $O(k \log k)$, the total time for sorting $\mathbf{B}_i$s is $O(mnb_{max} + (mn + b_{max} + \log k)k \log k)$.

In our experiments, we employ a more practical sorting network of Batcher's odd-even mergesort [36] of size $O(k \log^2 k)$. A simple oblivious radix sort [37] in the $O(k \log k)$

algorithm under the assumption that the bit length of each integer is constant was recently proposed.

3.1.3. Selecting Features

Let $(F_{\pi(1)}, \ldots, F_{\pi(k)})$ be the sorted list of features. We first compute a sequence of bit strings $(Z_2, \ldots, Z_k)$ of length mn each such that $Z_i[h] = \bigvee_{j=i+1}^{k} \mathbf{B}_{\pi(j)}[h]$ for any $2 \leq i \leq k$ and $1 \leq h \leq mn$; namely, Z_i is the bit array storing the cumulative or each position h for $\mathbf{B}_{\pi(i+1)}, \mathbf{B}_{\pi(i+2)}, \ldots, \mathbf{B}_{\pi(k)}$. Note that $Z_i[h] = 0$ indicates that the set $\{F_{\pi(i+1)}, F_{\pi(i+2)}, \ldots, F_{\pi(k)}\}$ of features is inconsistent with regard to a pair (x_p, y_q) satisfying $h = m(p-1) + q$, and $\{F_{\pi(i+1)}, F_{\pi(i+2)}, \ldots, F_{\pi(k)}\}$ is inconsistent if and only if the bit string Z_i contains 0. See Table 5 for Zs in our running example. The computation requires $O(kmn)$ time and space.

Table 5. Sorted **B**s for the example dataset D of Table 3 and the corresponding Z_is.

i	$\pi(i)$	$\mathbf{B}_{\pi(i)}$										Z_i									
1	2	$\mathbf{B}_2 = 1$	0	0	1	0	0	1	1	0	1	$Z_1 = 1$	1	1	1	1	1	1	1	1	1
2	4	$\mathbf{B}_4 = 0$	0	1	0	0	1	1	0	1	1	$Z_2 = 1$	1	1	1	1	1	1	1	1	1
3	3	$\mathbf{B}_3 = 0$	1	1	0	1	0	1	1	0	1	$Z_3 = 1$	1	0	1	1	1	1	0	1	1
4	1	$\mathbf{B}_1 = 1$	1	0	1	1	1	1	0	1	1	$Z_4 = 0$	0	0	0	0	0	0	0	0	0

We simulate Algorithm 1 on encrypted **B**s and Zs for feature selection. Furthermore, we use two 0-initialized bit arrays, R of length k and S of length mn. $R[i]$ is meant to store 1 if the i-th feature (in sorted order) is selected. S is used to keep track of the cumulative or for the bit strings of the currently selected features. Namely, $S[h]$ is set to $\bigvee_{\alpha=1}^{\ell} \mathbf{B}_{\pi(j_\alpha)}[h]$ if ℓ features $\{F_{\pi(j_1)}, \ldots, F_{\pi(j_\ell)}\}$ have been selected at this time.

Assume that we are in the i-th iteration of the loop of Algorithm 1. Note that, at this time, F contains features $\{F_{\pi(i)}, F_{\pi(i+1)}, \ldots, F_{\pi(k)}\}$ and currently selected features, and $F \setminus \{F_{\pi(i)}\}$ is consistent if $\bigwedge_{h=1}^{mn}(Z_i[h] \vee S[h])$ is 1. Because we keep $F_{\pi(i)}$ in F if $F \setminus \{F_{\pi(i)}\}$ is inconsistent, the algorithm sets $R[i] = \neg \bigwedge_{h=1}^{mn}(Z_i[h] \vee S[h])$. After computing $R[i]$, we can correctly update S by $S[h] \leftarrow S[h] \vee (R[i] \wedge \mathbf{B}_{\pi(i)}[h])$ for every $1 \leq h \leq mn$ in $O(mn)$ time. Therefore, the total computational time is $O(kmn)$.

3.1.4. Summing Up Analysis

The sorting step takes $O(mnb_{max} + (mn + b_{max} + \log k)k \log k)$ time. Because CWC works with any consistent measure, we do not need to use $\|\mathbf{B}_i\|$ in full accuracy, so we assume that b_{max} is set to be constant. Under the assumption, we obtain the following theorem.

Theorem 1. *For the two-party feature selection problem, we can securely simulate CWC in $O(kmn \log k + k \log^2 k)$ time and $O(kmn)$ space without revealing the private data of the parties under the assumption that TFHE is secure.*

Proof. According to the discussion above, computing $\mathbf{B}_i$ for all features takes $O(kmn)$ time and space, sorting features takes $O(mnb_{max} + (mn + b_{max} + \log k)k \log k) = O(kmn \log k + k \log^2 k)$ time, and selecting features takes $O(kmn)$ time.

Finally, party A computes in $O(k \log k)$ time an integer array P with $P[h] = R[h] \cdot \pi(h)$, which stores the original indices of selected features. In the outsourcing scenario, party A simply sends P to party B as the result of CWC. In the joint computing scenario, party A randomly shuffles P to conceal π to B. As a result, we can securely simulate CWC in $O(kmn \log k + k \log^2 k)$ time and $O(kmn)$ space. □

3.2. Improvement of Secure CWC

Sorting is a major bottleneck for private CWC. The reason for this is that pointers cannot be moved across ciphertexts. For example, consider the case of a secure integer. Let

the variables x and y contain integers a and b, respectively. In this case, by performing the secure operation $a <?b$, the result is obtained as $a <?b = c \in \{0,1\}$. Using this logical bit c, we can swap the values of x and y in $O(1)$ time, satisfying $x < y$ by the secure operation $x \leftarrow c \cdot a + \bar{c} \cdot b$ and $y \leftarrow \bar{c} \cdot a + c \cdot b$.

In the case of CWC, however, each integer i of feature F_i is associated with the bit string $\mathbf{B}_i$. Since any x cannot be decrypted, we cannot swap the pointers appropriately. Therefore, the baseline algorithm swaps $\mathbf{B}_i$ explicitly. As a result, the computation time for sorting increases to $O(mnk \log^2 k)$. Our main contribution is to improve this complexity to $O(mnk + k \log^2 k)$ by reducing the cost for such explicit sorting.

Based on the FHE, we propose the improved secure CWC (Algorithm 2), which reduces the time complexity to $O(mnk + k \log^2 k)$. An example run of Algorithm 2 is illustrated in Figure 1. As shown in this example, the party A can securely sort k randomized features in $O(k \log^2 k)$ time using a suitable *sorting network*, and then, according to the result of sorting, A swaps each associated bit string of length nm in $O(kmn)$ time. Following this preprocessing, the parties securely obtain minimal consistent features by decrypting the output of CWC. Finally, we have the following result.

Algorithm 2 Improved secure CWC between parties A and B

1: **Preprocessing:**
Party A has $E_{\mathsf{B}}[\mathcal{F}] = E_{\mathsf{B}}[\mathcal{F}_1, \ldots, \mathcal{F}_k]$ for $\mathcal{F}_i = (F_i, \|\mathbf{B}_i\|, \mathbf{B}_i)$ encrypted with party B's public key, where each datum x is encrypted at time 0 as $E_{\mathsf{B}}[x|0]$.
2: **Party A:**
Generates r_i for $i = 1, \ldots, n$ uniformly at random.
Sends $(E_{\mathsf{B}}[\mathbf{B}_i + r_i|1], E_{\mathsf{A}}[r_i|1])$ for $i = 1, \ldots, n$.
3: **Party A:**
Calculates $E_{\mathsf{B}}[i|2]$ for $i = 1, \ldots, n$.
Securely sorts $(E_{\mathsf{B}}[F_i|0], E_{\mathsf{B}}[\|\mathbf{B}_i\||0], E_{\mathsf{B}}[i|2])$ for $i = 1, \ldots, n$ in increasing order of $\|\mathbf{B}_i\|$.
As a result, obtains $(E_{\mathsf{B}}[F_{i_j}|3], E_{\mathsf{B}}[\|\mathbf{B}_{i_j}\||3], E_{\mathsf{B}}[i_j|3])$ for $j = 1, \ldots, n$.
Generates a permutation $\pi \in \mathfrak{S}_n$ uniformly at random and memorizes it.
Sends $(E_{\mathsf{B}}[i_{\pi(1)}|3], \ldots, E_{\mathsf{B}}[i_{\pi(n)}|3])$.
4: **Party B:**
Decrypts $(i_{\pi(1)}, \ldots, i_{\pi(n)})$.
Generates r'_i for $i = 1, \ldots, n$ uniformly at random.
Sends $(E_{\mathsf{B}}[\mathbf{B}_{i_{\pi(j)}} + r_{i_{\pi(j)}} + r'_{i_{\pi(j)}}|4], E_{\mathsf{A}}[r_{i_{\pi(j)}} + r'_{i_{\pi(j)}}|4])$ for $j = 1, \ldots, n$.
5: **Party A:**
Decrypts $r_{i_{\pi(j)}} + r'_{i_{\pi(j)}}$ for $j = 1, \ldots, n$.
Obtains $E_{\mathsf{B}}[\mathbf{B}_{i_{\pi(j)}}|5]$ $j = 1, \ldots, n$.
Obtains $E_{\mathsf{B}}[\mathbf{B}_{i_j}|5]$ $j = 1, \ldots, n$ through permutation by π^{-1}.
6: **Party A:**
Simulates CWC for resulting $E_{\mathsf{B}}[\mathcal{F}]$.

Theorem 2. *Algorithm 2 can simulate CWC in $O(kmn + k \log^2 k + k \log k \log mn)$ time and $O(kmn)$ space under the assumption that FHE executes each bit operation in $O(1)$ time.*

Proof. Compared to the baseline, additional space is required for π and r_i and r'_i. Thus, the space complexity remains $O(kmn)$. For the time complexity, the main task is to sort k-triple $(F_i, \|\mathbf{B}_i\|, \mathbf{B}_i)$ in the increasing order of $\|\mathbf{B}_i\|$. The improved algorithm sorts only the pairs $x_i = (F_i, \|\mathbf{B}_i\|)$ of integers, where the size of x_i is $O(\log k + \log mn)$ bits. For each x_i, x_j, we can check if $\|\mathbf{B}_i\| \leq \|\mathbf{B}_j\|$ in $O(\log mn)$ time and we can swap them in $O(\log k + \log mn)$ time using homomorphic operations in FHE. It follows that the time for sorting all x_i $(i = 1, \ldots, k)$ is $O(k \log k(\log k + \log mn))$ time. After sorting the pairs, the algorithm moves all $\mathbf{B}_i$ to the correct positions according to the rank of x_i $(i = 1, \ldots, k)$. This cost is $O(kmn)$. Therefore, time complexity is $O(kmn + k \log^2 k + k \log k \log mn)$. □

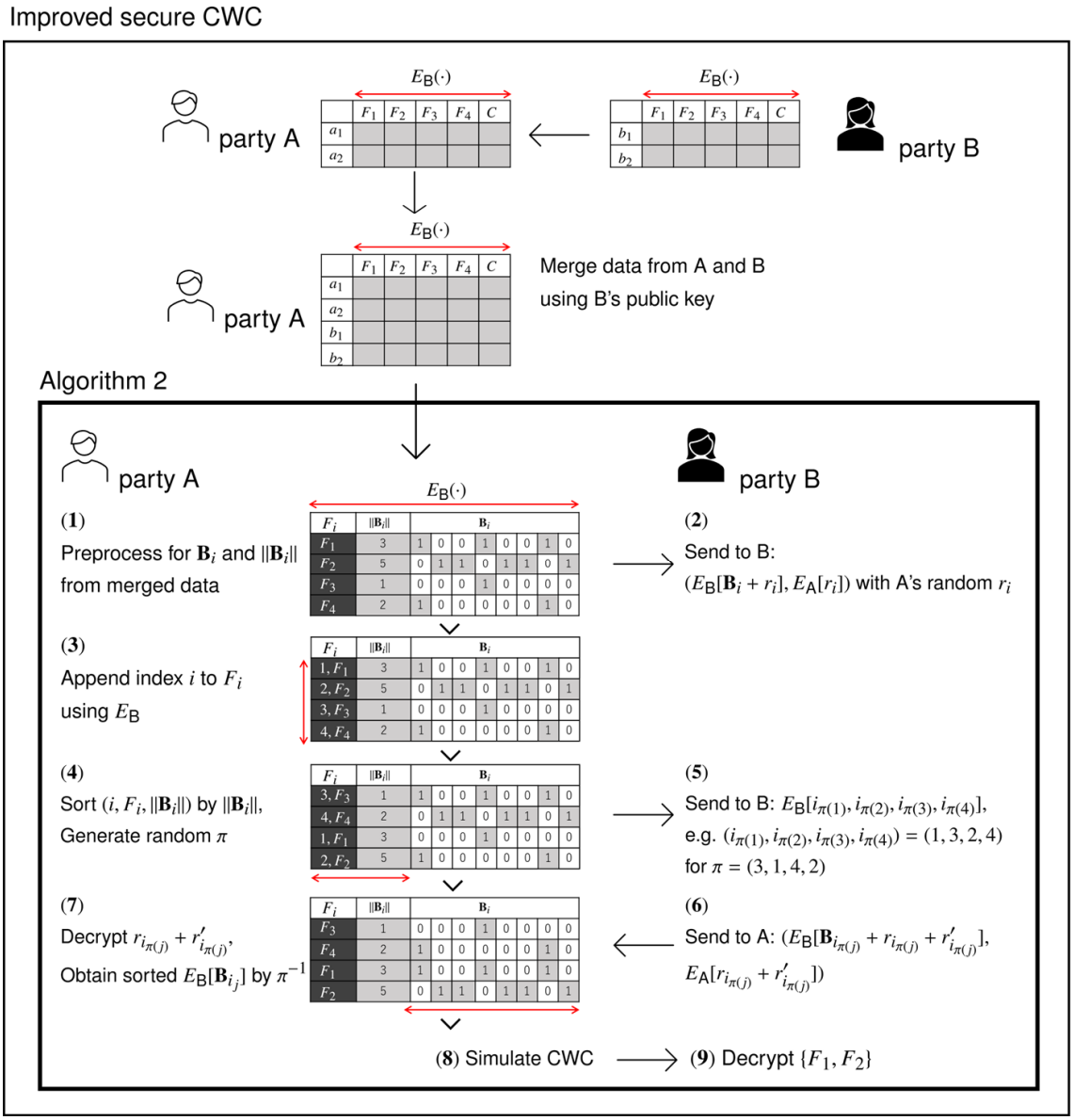

Figure 1. An example run of Algorithm 2. For simplicity, we omit the clock time in each ciphertext. (1): Parties A and B jointly compute $\mathbf{B}_i$ and $\|\mathbf{B}_i\|$ for each feature F_i (same as the baseline algorithm). (2): A securely sends $\mathbf{B}_i$; B cannot learn anything. (3): A appends encrypted index i for each F_i. (4): A sorts only $(F_i, \|\mathbf{B}_i\|)$ by $\|\mathbf{B}_i\|$. (5): A sends the sorted indices with random permutation; B cannot learn anything. (6): B sends $\mathbf{B}_i$; A cannot learn anything from it. (7): A decrypts the noise and obtains the correct order of $\mathbf{B}_{i_j}$; A cannot learn anything. (8): A simulates CWC the same as the baseline. (9): Party A, B share the resulting features.

Theorem 3. *Algorithm 2 is secure under the assumption that the employed FHE is IND-CPA secure.*

Proof. We show the security by constructing simulators for parties A and B, respectively.

B's view (what B can obtain from A) is the following:

- $(E_B[\mathbf{B}_i + r_i|1], E_A[r_i|1])$ for $i = 1, \ldots, n$;
- $E_B[i_{\pi(1)}|3], \ldots, E_B[i_{\pi(n)}|3]$.

Their probability distributions are uniform and independent of each other. Hence, the simulator for B can replace them with

- $(E_{\mathsf{B}}[\mathbf{B}_i + r''_i|6], E_{\mathsf{A}}[r''_i|6])$ for $i = 1, \ldots, n$ and r''_i, which are selected uniformly at random;
- $E_{\mathsf{B}}[\pi'(1)|6], \ldots, E_{\mathsf{B}}[\pi'(n)|6]$ for $\pi' \in \mathfrak{S}_n$, which is selected uniformly at random.

Note that, even if an adversary knows $\mathbf{B}_i$, it is computationally impossible to distinguish between $E_{\mathsf{B}}[\mathbf{B}_i + r''_i|6]$ and $E_{\mathsf{B}}[r''_i|6]$ by the IND-CPA security of the cryptosystem E_{B}.

Next, we construct a simulator `Sim` for the party A. Although what A can obtain from B is

$$\left\{ \left(E_{\mathsf{B}}[\mathbf{B}_{i_{\pi(j)}} + r_{i_{\pi(j)}} + r'_{i_{\pi(j)}}|4], E_{\mathsf{A}}[r_{i_{\pi(j)}} + r'_{i_{\pi(j)}}|4] \right) \middle| j = 1, \ldots, n \right\}$$

this is equivalent to $\{E_{\mathsf{B}}[\mathbf{B}_{i_j}|5] \mid j = 1, \ldots, n\}$ after decryption and permutation.

On the other hand, the sequence $(i_1, \ldots, i_n)$ is not explicitly given to A, and A recognizes it through the alignment between

- $(E_{\mathsf{B}}[\|\mathbf{B}_{i_1}\||3], \ldots, E_{\mathsf{B}}[\|\mathbf{B}_{i_n}\||3])$ and
- $(E_{\mathsf{B}}[\mathbf{B}_{i_1}|5], \ldots, E_{\mathsf{B}}[\mathbf{B}_{i_n}|5])$.

Therefore, we define A's view to be

$$\mathtt{View}_{\mathsf{A}} = \left\{ \left(E_{\mathsf{B}}[\|\mathbf{B}_{i_j}\||3], E_{\mathsf{B}}[\mathbf{B}_{i_j})|5] \right) \middle| j = 1, \ldots, n \right\}$$

with $\|\mathbf{B}_{i_1}\| \leq \cdots \leq \|\mathbf{B}_{i_n}\|$.

On the other hand, we define the view that `Sim` should generate as follows. While A can generate $\{E_{\mathsf{B}}[\|\mathbf{B}_{i_j}\||3] \mid j = 1, \ldots, n\}$ with $\|\mathbf{B}_{i_1}\| \leq \cdots \leq \|\mathbf{B}_{i_2}\|$, A needs B's cooperation to generate $\{E_{\mathsf{B}}[\mathbf{B}_{i_j}|5] \mid j = 1, \ldots, n\}$. Without B's cooperation, `Sim` selects $\pi' \in \mathfrak{S}_n$ uniformly at random, and generates its own view to be

$$\mathtt{View}_{\mathtt{Sim}} = \left\{ \left(E_{\mathsf{B}}[\|\mathbf{B}_{i_j}\||3], E_{\mathsf{B}}[\mathbf{B}_{\pi''(j)}|*] \right) \middle| j = 1, \ldots, n \right\}.$$

`Sim` can compute $E_{\mathsf{B}}[\mathbf{B}_{\pi''(j)}|*]$ from $E_{\mathsf{B}}[0|*]$ and $E_{\mathsf{B}}[\mathbf{B}_{\pi''(j)}|0]$ taking advantage of the homomorphic property of the encryption system E_{B}.

Furthermore, we define a distinguisher $\mathcal{D}$ as a PPT Turing machine that tries to distinguish between $\mathtt{View}_{\mathsf{A}}$ and $\mathtt{View}_{\mathtt{Sim}}$ on the input of $\{(E_{\mathsf{B}}[\|\mathbf{B}_i\||0], E_{\mathsf{B}}[\mathbf{B}_i|0]) \mid i = 1, \ldots, n\}$.

When we let $\Pr[Y = \mathsf{A} \mid X = \mathsf{A}] = 1/2 + \alpha_1$ and $\Pr[Y = \mathtt{Sim} \mid X = \mathtt{Sim}] = 1/2 + \alpha_2$, the advantage of $\mathcal{D}$ is defined as $\alpha_1 + \alpha_2$.

We show that, if $\mathcal{D}$'s advantage α is not negligible, we can construct a PPT attacker `Attck` that can break the IND-CPA security of the encryption system E_{B} with a non-negligible advantage. Our attacker `Attck` plays the IND-CPA game, exploiting an oracle $\mathcal{O}_{\mathrm{IND}}$ as follows:

1. `Attck` generates $\mathbf{B}_1, \mathbf{B}_2$ with $\|\mathbf{B}_1\| \leq \|\mathbf{B}_2\|$;
2. `Attck` lets $x_1 = \|\mathbf{B}_1\|$ and $x_2 = \|\mathbf{B}_2\|$ and throws a query (x_1, x_2) to $\mathcal{O}_{\mathrm{IND}}$;
3. $\mathcal{O}_{\mathrm{IND}}$ selects $i \in \{1, 2\}$ uniformly at random and sends $c = E_{\mathsf{B}}[x_i| - 1]$ to `Attck`;
4. `Attck` initializes $\mathcal{D}$ by inputting $(E_{\mathsf{B}}[\|\mathbf{B}_1\||0], E_{\mathsf{B}}[\|\mathbf{B}_1\||0]), (E_{\mathsf{B}}[\|\mathbf{B}_2\||0], E_{\mathsf{B}}[\|\mathbf{B}_2\||0])$;
5. **First query.** `Attck` throws to $\mathcal{D}$ the query: $(E_{\mathsf{B}}[\|\mathbf{B}_2\||2], c), (E_{\mathsf{B}}[\|\mathbf{B}_2\||2], E_{\mathsf{B}}[\mathbf{B}_2|2])$;
6. If $\mathcal{D}$ replies with A, `Attck` outputs 1 and terminates.
7. **Second query.** `Attck` generates c' by adding $E_{\mathsf{B}}[0|3]$ to c. Note that $\mathcal{D}_{\mathsf{B}}(c') = \mathcal{D}_{\mathsf{B}}(c)$ holds. `Attck` throws to $\mathcal{D}$ the query: $(E_{\mathsf{B}}[\|\mathbf{B}_1\||3], E_{\mathsf{B}}[\mathbf{B}_2|3]), (E_{\mathsf{B}}[\|\mathbf{B}_2\||3], c')$.
8. If $\mathcal{D}$ replies with `Sim`, `Attck` outputs 1 and terminates.
9. `Attck` outputs 2.

We evaluate `Attck`'s advantage as follows. We assume $\mathcal{D}_{\mathsf{B}}(c) = x_1$. The probability of this case is $1/2$. The probability that $\mathcal{D}$ replies with A to the first query or $\mathcal{D}$ replies with `Sim` to the second query is

$$\frac{1}{2} + \alpha_1 + \frac{1}{2} + \alpha_2 - (\frac{1}{2} + \alpha_1)(\frac{1}{2} + \alpha_2) = \frac{3}{4} + \frac{\alpha}{2} - \alpha_1\alpha_2 \geq \frac{3}{4} + \frac{\alpha}{4},$$

since the first and second queries are mutually independent.

When assuming $D_B(c) = x_2$, we see that $\Pr[\mathcal{D}$ outputs `Sim` at the first query $] - 1/2$ is negligible. Otherwise, $\mathcal{D}$ can be used as an attacker to break the IND-CPA security of E_B. Therefore, $\Pr[$`Attck` outputs $2] - 1/4$ is negligible. Consequently, we have

$$\Pr[\texttt{Attck's guess is right}] \geq \frac{1}{2}\left(\frac{3}{4} + \frac{\alpha}{4}\right) + \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{2} + \frac{\alpha}{8}.$$

Since we assume that α is not negligible, neither is $\alpha/4$. □

4. Experiments

We implemented the baseline and improved algorithms for secure CWC in `C++` using the TFHE library (https://tfhe.github.io/tfhe (accessed on 28 January 2021)).

The experiments were carried out on a machine equipped with an Intel Core i7-6567U (3.30 GHz) processor and 16GB of RAM. In the following, m (resp. n) is the number of positive (resp. negative) data and k is the number of features.

Table 6 summarizes the running time of the baseline algorithm (naive implementation of Algorithm 1 using TFHE) for random data generated for $k \in \{10, 50, 100\}$ and $mn \in \{100, 500, 1000\}$. The complexity analysis shows that the running time increases in proportion to mn. This experimental result confirms this in real data. The table clearly shows that the sorting process is the bottleneck.

Table 6. Running time (s) of baseline algorithm (naive secure CWC). Task 1: computing $\mathbf{B}_i$s. Task 2: sorting $\mathbf{B}_i$s. Task 3: feature selection.

k	mn	Task 1	Task 2	Task 3
10	100	60.3	**835.8**	111.9
	500	300.5	**4252.4**	558.1
	1000	601.4	**8867.0**	1114.2
50	100	301.8	**6292.6**	589.3
	500	1502.9	**30,364.6**	2941.0
	1000	3007.0	**62,124.6**	5919.7
100	100	603.7	**16,148.5**	1179.0
	500	3005.9	**76,315.2**	5952.5
	1000	6014.1	**154,143.5**	11,867.0

Table 7 compares the running time of preprocessing in the baseline and improved algorithms. According to the results, the proposed algorithm significantly improves the bottleneck in naive CWC for secure computing. We should note that the baseline and improved algorithms both compute exactly the same solution as the CWC on plaintexts. We also show the details of the improved algorithm: 'sorting' means the time for sorting of the triples $(F_i, ||\mathbf{B}_i||, i)$ of integers; 'other task' means the time for remaining tasks, including generating/adding/subtracting random noise r_i, moving $\mathbf{B}_i$, decrypting integers, etc.

Table 7. Running time (s) of baseline and improved algorithms. Here, 'baseline' is same as Task 2 in Table 6 (i.e., the bottleneck); 'improved:' is the running time of corresponding task in the improved algorithm, where 'sorting' and 'other tasks' are the details.

k	mn	Baseline	Improved:	Sorting	Other Tasks
10	100	**835.8**	**203.7**	69.4	134.2
	500	**4252.4**	**286.2**	89.5	196.6
	1000	**8867.0**	**302.5**	98.9	203.6
100	100	**16,148.5**	**3311.2**	1865.5	1445.7
	500	**76,315.2**	**4601.9**	2647.7	1954.1
	1000	**154,143.5**	**4671.4**	2660.8	2010.5

Table 8 displays the running time of the improved algorithm for real data available from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/index.php (accessed on 26 January 2022)), because, since these datasets contain more than three feature/class values, we treated them as a binary classification between one feature/class and the other.

Table 8. Running time (s) of improved algorithm for real data in UCI Machine Learning Repository.

Dataset	k	mn	Time	Sorting	Other Tasks
`Letter`	16	196	**252.2**	80.6	171.4
`Breast Cancer`	10	2464	**312.6**	103.1	209.5
`Covertype`	54	979	**1653.5**	836.9	816.6

We demonstrated that the proposed algorithm works well for real-world multi-level feature selection problems. We only evaluated the running time in this experiment, but the relevance of the extracted features is guaranteed because the secure CWC algorithm produces the same solution as the original [8].

5. Conclusions

On the basis of fully homomorphic encryption, we proposed a faster private feature selection algorithm that allows us to securely compute functional features from distributed private datasets. Our algorithm can simulate the original CWC algorithm, which chooses favorable features by sorting. In addition to the improvement in computational complexity, the proposed algorithm solves the private feature selection problem in practical time for a variety of real data. One of the remaining challenges is to improve sorting at a lower cost because CWC does not always require exact sorting. Then, ambiguous sorting possibly reduces the computation time, maintaining solution quality. At this time, the proposed algorithm is not applicable to real numbers for feature value. This is because TFHE [21] is not suitable for floating-point operations. Extending the TFHE library to enable secure feature selection for real-valued data is a future challenge.

A well-known feature selection method is to filter features by computing *Gini impurity* scores [14]. In this method, the optimal threshold for filtering is determined by the order of sorting of each feature. However, since sorting is time-consuming even for secret sharing-based MPC [38], a simpler method of determining the threshold by calculation has been proposed [30], and its effectiveness has been confirmed by experiments. On the other hand, this study focuses on consistency measure-based feature selection. As we mentioned previously, the consistency measure-based method has been confirmed to have advantages over other methods. This study proposes the first secure protocol that enables consistency measure-based feature selection in practical time.

We next compare our proposal with other methods in the framework of private feature selection. A secret sharing-based MPC using the distributed secure sum is proposed [29]. In this method, it is known that statistical information about the data is leaked during the computation. The authors in [30] propose an honest-majority three-party protocol that improves on the drawback of [29]. This method is fast but requires at least two trusted parties. In addition, [30] considered both semi-honest and malicious adversaries, but in this study, the parties are assumed to be semi-honest. A feature selection using homomorphic encryption was proposed in [26]. This method uses only the additive homomorphic property in a two-party model, which limits its computational power. Therefore, statistical information about the data is leaked because partial decryption is required during communication between the parties. Moreover, this protocol has not been implemented. The recent approach by [39] is not based on cryptography and does not provide a formal privacy guarantee, and it leaks information through the disclosure of intermediate representations. Although our method is inferior to secret sharing-based MPC in terms of practical computing time, it

can handle feature selection from many data owners, even in situations where only they themselves can be trusted, and does not leak information during computation.

In addition, we discuss future issues and prospects related to secret computation with FHE, which were not discussed in detail in this study. First, this study assumes that the data are consistent, which cannot be applied to real-world data. If the data are inconsistent, e.g., when the data are merged, there are two entries with the same feature values they but are classified into different classes, the party can ignore these entries without decoding. Since the outsourced party can perform a comparison of two integers without decoding, they can use the encrypted logical bit to change the class label of these irrelevant entries to a special value, effectively ignoring them.

Next, we consider how to speed up the computation of real numbers using FHE. CKKS [40] and TFHE are the current state-of-the-art methods for computing real numbers on FHE. CKKS speeds up arithmetic operations on real numbers by converting reals to integers through scaling, performing arithmetic operations on the integers, and then converting the results back to reals. However, CKKS has the drawback that it cannot compute nonlinear functions (e.g., ReLU) or perform comparison operations, making it difficult to apply to ML. On the other hand, TFHE can evaluate comparison operations and NAND circuits, making all computations theoretically possible. Unfortunately, the runtime on TFHE has an overhead of approximately 10,000 times that on the plaintext, and thus the GPGPU-based architecture is currently being studied for speedup. Currently, the fastest implementation is around 20 times faster than algorithms on CPUs [41]. Thus, the application and speedup of TFHE to ML is a promising research area for the future.

In conclusion, it should be noted that with the development of FHE, practical algorithms for more challenging problems such as large-scale genome analysis (GWAS) [42,43] and deep learning [44–46] are emerging.

Author Contributions: Conceptualization, H.S.; methodology, T.I., K.S. and H.S.; software, S.O., J.T. and M.K.; validation, S.O., J.T. and M.K.; formal analysis, T.I., K.S. and H.S.; investigation, T.I., K.S. and H.S.; resources, S.O., J.T. and M.K.; data curation, S.O., J.T. and M.K.; writing—original draft preparation, T.I., K.S. and H.S.; writing—review and editing, T.I., K.S. and H.S.; visualization, H.S.; supervision, H.S.; project administration, H.S.; funding acquisition, 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 21H05052, 18H04098).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Wang, S.; Zhang, Y.; Dai, W.; Lauter, K.; Kim, M.; Tang, Y.; Xiong, H.; Jiang, X. HEALER: Homomorphic computation of ExAct Logistic rEgRession for secure rare disease variants analysis in GWAS. *Bioinformatics* **2015**, *32*, 211–218. [CrossRef] [PubMed]
2. 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.
3. Qiu, G.; Huo, H.; Gui, X.; Dai, H. Privacy-Preserving Outsourcing Scheme for SVM on Vertically Partitioned Data. *Secur. Commun. Netw.* **2022**, *2022*, 9983463. [CrossRef]
4. Bost, R.; Ada Popa, R.; Tu, S.; Goldwasser, S. Machine learning classification over encrypted data. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 8–11 February 2015.
5. Khedr, A.; Gulak, G.; Vaikuntanathan, V. SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers. *IEEE Trans. Comput.* **2016**, *65*, 2848–2858. [CrossRef]
6. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014** *40*, 16–28. [CrossRef]
7. Shin, K.; Kuboyama, T.; Hashimoto, T.; Shepard, D. SCWC/SLCC: Highly scalable feature selection algorithms. *Information* **2017**, *8*, 159. [CrossRef]
8. Shin, K.; Xu, X.M. Consistency-based feature selection. In Proceedings of the 13th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Santiago, Chile, 28–30 September 2009; pp. 28–30.

9. Almuallim, H.; Dietteric, T.G. Learning boolean concepts in the presence of many irrelevant features. *Artif. Intell.* **1994**, *69*, 279–30. [CrossRef]
10. 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.
11. 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.
12. 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.
13. De Cock, M.; Dowsley, R.; Nascimento, A.C.A.; Railsback, D.; Shen, J.; Todoki, A. High performance logistic regression for privacy-preserving genome analysis. *BMC Med. Genom.* **2021**, *14*, 23. [CrossRef] [PubMed]
14. Breiman, L.; Friedman, J.; Stone, C.; Olshen, R. *Classification and Regression Trees*, 1st ed.; Taylor and Francis: Oxfordshire, UK, 1984.
15. 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.
16. Attrapadung, N.; Hanaoka, G.; Mitsunari, S.; Sakai, Y.; Shimizu, K.; Teruya, T. Efficient two-level homomorphic encryption in prime-order bilinear groups and a fast implementation in webassembly. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Korea, 4 June 2018; pp. 685–697.
17. 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.
18. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. In Proceedings of the 3rd Innovations in Theoretical Computer Science, Cambridge, MA, USA, 8–10 January 2012; pp. 309–325.
19. 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.
20. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: Fast fully homomorphic encryptionover the torus. *J. Cryptol.* **2020**, *33*, 34–91. [CrossRef]
21. 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).
22. 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.
23. Goldwasser, S.; Micali, S. Probabilistic Encryption. *J. Comput. Syst. Sci.* **1984**, *28*, 270–299. [CrossRef]
24. Fujisaki, E.; Okamoto, T.; Pointcheval, D.; Stern, J. RSA-OAEP is secure under the RSA assumption. In Proceedings of the 21st Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 260–274.
25. Bellare, M.; Rogaway, P. Optimal Asymmetric Encryption. In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, 9–12 May 1994; pp. 92–111.
26. Rao, V.; Long, Y.; Eldardiry, H.; Rane, S.; Rossi, R.A.; Torres, F. Secure two-party feature selection. *arXiv* **2019**, arXiv:1901.00832.
27. Anarakia, J.R.; Samet, S. Privacy-preserving feature selection: A survey and proposing a new set of protocols. *arXiv* **2020**, arXiv:2008.07664.
28. Banerjee, M.; Chakravarty, S. Privacy preserving feature selection for distributed data using virtual dimension. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, Glasgow Scotland, UK, 24–28 October 2011; pp. 2281–2284.
29. Sheikhalishahi, M.; Martinelli, F. Privacy-utility feature selection as a privacy mechanism in collaborative data classification. In Proceedings of the 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, Poznan, Poland, 21–23 June 2017; pp. 244–249.
30. 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.
31. Abspoel, M.; Escudero, D.; Volgushev, N. Secure training of decision trees with continuous attribute. *Proc. Priv. Enhancing Technol.* **2021**, *2021*, 167–187. [CrossRef]
32. Dash, M.; Liu, H. Consistency-based search in feature selection. *Articial Intell.* **2003**, *151*, 155–176. [CrossRef]
33. Pawlak, Z. *Rough Sets, Theoretical Aspects of Reasoning about Data*; Kluwer Academic Publishers: Alphen aan den Rijn, The Netherlands, 1991.
34. Arauzo-Azofra, A.; Benitez, J.M.; Castro, J.L. Consistency measures for feature selection. *J. Intell. Inf. Syst.* **2008**, *30*, 273–292. [CrossRef]
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.
36. 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.

37. Hamada, K.; Chida, K.; Ikarashi, D.; Takahashi, K. Oblivious Radix Sort: An Efficient Sorting Algorithm for Practical Secure Multi-party Computation (iacr.org). 2014. Available online: https://eprint.iacr.org/2014/121 (accessed on 26 January 2022).
38. Goodrich, M. Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in $O(nlogn)$ time. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, New York, NY, USA, 31 May–3 June 2014; pp. 684–693.
39. Ye, X.; Li, H.; Imakura, A.; Sakurai, T. Distributed collaborative feature selection based on intermediate representation. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 4142–4149.
40. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Online, 30 November 2017; pp. 409–437.
41. Matsuoka, K.; Hoshizuki, Takashi Sato, T.; Bian, S. Towards Better Standard Cell Library: Optimizing Compound Logic Gates for TFHE. In Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, New York, NY, USA, 15 November 2021; pp. 63–68.
42. Bos, J.W.; Lauter, K.; Naehrig, M. Private predictive analysis on encrypted medical data. *J. Biomed. Inform.* **2014**, *50*, 234–243. [CrossRef]
43. Lauter, K.; López-Alt, A.; Naehrig, M. Private Computation on Encrypted Genomic Data. In Proceedings of the 3rd International Conference on Cryptology and Information Security in Latin America, Florianópolis, Brazil, 17–19 September 2014; pp. 3–27.
44. Dowlin, N.; Gilad-Bachrach, R.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. CryptoNets: Applying neural networks to Encrypted data with high throughput and accuracy. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 201–210.
45. Bourse, F.; Minelli, M.; Minihold, M.; Paillier, P. Fast Homomorphic Evaluation of Deep Discretized Neural Networks. In Proceedings of the 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2018; pp. 483–512.
46. Badawi, A.A.; Jin, C.; Lin, J.; Fook Mun, C.; Jun Jie, S.; Hong Meng Tan, B.; Nan, X.; Mi Mi Aung, K.; Chandrasekhar, V.R. Towards the AlexNet Moment for Homomorphic Encryption: HCNN, the First Homomorphic CNN on Encrypted Data With GPUs. *IEEE Trans. Emerg. Top. Comput.* **2021**, *9*, 1330–1343. [CrossRef]