

## Article

# Chain-of-Blocks Assisted Secure Feature Selection, Federated Learning and Classifications in Cloud and Distributed Malicious Edge IoT Environments

Artrim Kjamilji

Faculty of Engineering and Natural Sciences, Department of Computer Engineering, Istanbul Sabahattin Zaim University, Halkalı Merkez, Halkalı, Küçükçekmece, 34303 Istanbul, Turkey; artrim.kjamilji@izu.edu.tr

## Abstract

We tackle the problem of secure and private feature selection by homomorphically evaluating features' information gains over the encrypted data of horizontally partitioned private datasets owned by edge IoT (Internet of Things) devices. In the process, we use a powerful cloud server to do the bulk of the costly homomorphic encryption aggregations. We proceeded with secure and private federated learning (training) and Machine Learning (ML) classification over the selected features in the same environmental settings (context). In the process, the participants interact with each other under strict security, privacy, and efficiency requirements. To this end, to each participant's interaction we provide confidentiality, integrity, and authenticity (CIA) by signing its hashed contents with the corresponding participant's private key. We assure consistency among interactions by introducing timestamps and linking them with the hashed content(s) of the preceding interaction(s). Those linked blocks of hashed content(s) from each interaction of participants while running the protocols produce the so-called chain-of-blocks (COB) structure, which will be utilized to detect malicious edge IoT dataset owners, unauthorized participants, and network errors. The security of the proposed protocols is proven through rigorous mathematical modeling. Extensive experimental evaluations over benchmark datasets give an advantage to our secure protocols ranging from several times to orders of magnitudes w.r.t to the state of the art in terms of computation and communication costs, as well as security and privacy characteristics. Moreover, since the utilized underlying cryptographic techniques are resilient to quantum computer attacks, the proposed algorithms are applicable to the post-quantum world.

**Keywords:** chain-of-blocks (COB); machine learning; secure IoT; privacy preserving; feature selection; federated learning; training; classification; homomorphic encryption; information gain; text classification; fault tolerance; collusion attacks; distributed environments; post-quantum cryptography; cloud computing; edge computing

Academic Editors: Suo Gao and Aditya Kumar Sahu

Received: 15 April 2026

Revised: 25 May 2026

Accepted: 28 May 2026

Published: 30 June 2026

**Copyright:** © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

## 1. Introduction and Problem Statement

Acknowledging the privacy concern while dealing with Machine Learning (ML) and Data Mining (DM) algorithms, the works in [1,2] appeared almost simultaneously and they were among the first to address the privacy of the datasets over which ML models are trained. Soon, other works followed, some of which are presented in the surveys in [3]

and [4]. Those are known as **privacy-preserving (PP) training** schemes. Common for them was that they solely concentrated on the privacy of the datasets, with little or no concern for the privacy of the classification stage. They also paid no attention to the privacy of the trained model [3,4]. Similar privacy-protection techniques were also observed in [5–10].

Realizing these shortcomings, dozens of schemes followed afterwards, which exclusively deal with the classification (inference) stage for different ML classifiers such as Naïve Bayes (NB), Deep Neural Networks, Support Vector Machines, Logistic Regression, Random Forests, Decision Trees, etc. These schemes fall into the **privacy-preserving classification** category, such as those in [11–25], to name a few. There are also a few schemes that, for consistency and continuity reasons, address both privacy-preserving training and classification problems under the same system architecture and environmental settings, such as those proposed in [26–32].

However, before training an ML model, it is usually common practice to do some pre-processing and feature selection for the dataset(s), over which the training is being done. Especially, this is common for the highly dimensional datasets (i.e., have a huge number of features), a typical occurrence for textual datasets such as SMS spam, e-mail spam, document classification, etc., which are known to harbor tens, hundreds of thousands, or even millions of unique features (words, tokens, bigrams, etc.). By applying some feature selection over such datasets, not only do we save valuable amounts of computation and communication costs during the training and classification stages later, but rather trained models also obtained over datasets on which a certain feature selection has been applied are shown to be more accurate during the classification stage [33–40].

Despite this, surprisingly, there is little work done on the area of secure and private feature selection, especially on edge Internet of Things (IoT) environments. The few existing schemes have one or several setbacks and disadvantages, which are elaborated on in Section 2. Among others, they are not designed to provide a mechanism for ensuring the end-to-end integrity, authenticity, and consistency (continuity, order of sequence) by which the interactions are done among different participants in the system while executing the protocols, nor are they designed to address malicious users that deviate from the protocols. Due to several definitions in the literature, to avoid eventual confusions, we give what we feel are the proper definitions over which we built our secure and private protocols.

**Definition 1.** A *chain-of-blocks (COB)* is a data log structure consisting of a list of blocks (log records), where each block consists of a timestamp and its transaction data (usually in form of a Merkle tree [41]) and is linked with the previous block(s) by including its (their) hash(es) in itself, thus forming a chain-of-blocks. Chain-of-blocks (COB) data structure is inspired by blockchain cryptocurrency technologies [33,42]. Details are provided in Section 3.4.

**Definition 2.** *Internet of Things (IoT)* is a set of cooperating devices (sensors, mobile devices, etc.) that can collect and transfer data over a wireless network without explicit human intervention with the aim of implementing a certain functionality [43].

**Definition 3.** *Edge computing* refers to the technologies that enable computations to be performed at the edge (end) of the network. It is usually done to improve response times and use less network resources, as well as resources of other devices around, by doing local processing at the edge node (device). For example, smartphones, laptops, or tablet PCs are edge devices [44].

**Paper contributions:** To overcome the afore-mentioned shortcomings (elaborated in more details in Section 2), this paper introduces the following six contributions:

1. **Secure feature selection (*secFS*) protocol**—novel secure feature selection (filtering) protocol based on information theoretic metrics such as entropy. Concretely, homomorphically, we evaluate features' information gains **over horizontally partitioned private datasets owned by edge IoT devices** and select the top  $m$  ones with the highest information gain. To the best of the author's knowledge, this is the first scheme in the literature to apply secure feature selection on information gains over distributed datasets. Its privacy requirements are as follows (details in Section 5):
  - **Privacy of the input features (words, tokens, attributes, bigrams)**, which are not public (e.g., words in a private email dataset), by permuting the hash bits of each feature.
  - **Privacy of the input feature values (counts, frequencies).**
  - **Privacy of intermediate results** since they might be used as a trapdoor for a total or partial leakage of the input or the output of the corresponding protocol.
  - **Privacy for the output**, which are the top  $m$  selected features with the highest information gain. They are kept private for the secure feature selection (*secFS*) and secure federated learning (training) (*sect*) protocols but are revealed only for the secure classification protocol (*secC*) since users prepare their queries according to selected features.
2. **Secure Federated Learning (Training) (*sect*) protocol**—on top of the selected (filtered)  $m$  features, we provide a **secure and private federated learning (training)** over the same context (system architecture, participants, and security settings). Security requirements of the federated learning protocol are as follows Section 6):
  - **Privacy of the input features**, which are the  $m$  selected features, i.e., the output from *secFS* protocol;
  - **Privacy of the input features' values;**
  - **Security and privacy of intermediate results;**
  - **Privacy of the output, i.e., the trained ML model.** This is one of the rare protocols in the literature to keep private the final trained ML model at any stage.
3. **Secure ML classifications (*secC*) protocol: to classify unclassified client queries using the obtained ML trained model** while having in mind the following (Section 7):
  - **Privacy of the trained ML model**, which will be used to classify user queries;
  - **Privacy of the user (client) query**, both query features and their values;
  - **Security and privacy of the intermediate results;**
  - **Privacy of the output, i.e., the final ML classification result;**
4. **Introduction of the chain-of-blocks log record structure**—For each participant's interaction, we ensure confidentiality, integrity, and authenticity (CIA) by digitally signing the hashed input content using the participant's corresponding private key. Consistency across interactions is maintained through the incorporation of timestamps and by cryptographically linking each interaction to the hashed content of the preceding interaction(s), thus building the structure that we call chain-of-blocks (COB). Details are in Section 3.4.
5. **Dealing with malicious edge IoT dataset owners and system errors by utilizing the chain-of-blocks:** After obtaining the trained ML model, ML prediction accuracy is tested. A low accuracy indicates contributions from malicious edge IoT devices, which are flagged as such. Hashes of COB are used to check network transmission errors, signatures on blocks of COB check the authorization of participants and timestamps on blocks of COB make sure that order of execution is correct. The details are given in Sections 3.4 and 8.
6. Extensive experimental evaluations show that the proposed protocols outperform the related ones in literature in terms of computation and communication costs as well as

on security and privacy characteristics. More details are provided in Sections 2, 9, and 10.

## 2. Related Research and State-of-the-Art

Early secure feature selection schemes rely mostly on the *secure sum* protocol [34–38] as their building block. In *secure sum*,  $n$  participants, denoted as  $P_1, \dots, P_n$ , securely compute the sum of their corresponding private integer inputs,  $i_1, \dots, i_n$ , assuming an existence of a ring network topology between them. Without the loss of generality, the first participant adds a random number  $R$  to his input and passes it to the second one, which in turn adds his private input to the received sum and passes the result to the third participants, and so on, until the first participant receives the randomized sum  $(R + \sum_{i=1}^n i_i)$ , subtracts the random  $R$  from it, and broadcasts the result to all the other participants. In this sense, while executing the protocol,  $P_t$  receives  $(R + \sum_{i=1}^{t-1} i_i)$  from  $P_{t-1}$  and after adding its private input transmits  $(R + \sum_{i=1}^t i_i)$  to  $P_{t+1}$ , where  $1 \leq t \leq n$ . This makes the communication cost of *secure sum* to be  $n \cdot \sum f$  transmissions and  $\sum f$  broadcasts, where  $\sum f$  is the total number of feature values for which we have to find the sum over  $n$  participants. If  $P_{t-1}$  and  $P_{t+1}$  collude, they can retrieve  $P_t$ 's private input  $i_t$  by subtracting  $P_t$ 's output and input, thus  $(R + \sum_{i=1}^t i_i) - (R + \sum_{i=1}^{t-1} i_i) = i_t$ . An eavesdropper that can listen to all communicating channels can retrieve the private input of all the participants. Another drawback of *secure sum* is that it suffers from high communication overhead when  $f$  is large, which is the case with the highly dimensional text classification datasets, thus making *secure sum* highly impractical. Furthermore, *secure sum* does not work in the secure two-party computation (2PC) scenarios.

Among the first schemes to address the problem of secure feature selection is [34]. It calculates the features' misclassification gain [34] by proposing a probabilistic protocol based on *secure sum*, which does not always guarantee avoidance of the collusion attack, despite increasing the communication cost. In [35], they use the Harsanyi–Farrand–Chang [35] metric for feature selection, which, for each feature value, needs a few invocations of *secure sum* to find intermediate metrics such as correlation or covariance, without addressing any of the drawbacks of *secure sum*. In [36], they test several metrics for feature selection and, in the process, provide a trade-off between privacy and the accuracy of the trained model. In [37], each participant splits his private input into  $n$  shares such that their sum is equal to the participant's private input and sends one share to each of the other participants. Afterwards, all the participants locally sum up the shares received from others and invoke *secure sum* to find the final result. While this solves the collusion attack, it introduces an overhead of  $n \cdot (n - 1)$  transmissions into the high communication cost of a single *secure sum* invocation. After giving a brief literature review on the topic, ref. [38] proposes ideas for a few secure feature selection schemes without engaging in implementation details. Among others, ref. [38] inherits all the drawbacks of *secure sum* since it is supposed to use it as its main building block.

None of [34–38] solved the *secure sum*'s shortcomings of an eavesdropper that can intercept all the communications, of the inability to deal with 2PC, and of the high communication overhead, especially knowing that the metrics that they propose need several invocations of *secure sum* for a single feature value. In this context, [34–38] use a total of several hundreds of thousands of rounds (interactions) compared to only a few that our protocol uses to reach the same goal with textual datasets.

Avoiding *secure sum*, ref. [39] uses Paillier's additive homomorphic encryption scheme [45] to introduce a four-round 2PC (relatively high for a 2PC in secure ML classification scenarios) and does not address the scenario with multiple datasets. The secure feature-selection scheme in [40] uses secret sharing for multiple dataset owners to share their data among three non-colluding servers (3PC), a scenario that is not easily feasible

in reality, especially knowing that even if two of them collude, it can reveal the private dataset input values. Furthermore, ref. [40] has a high interaction rate, and knowing that each feature value is processed independently makes it have a high communication overhead. Depending on to whom the trained model is shown, ref. [40] can also deal with two dataset owners when the owner of the trained model is not one of the dataset owners, which rarely happens in practice.

Even if they do not engage in secure feature selection, we consider the schemes that deal with privacy preserving training and/or classifications of textual datasets, such as SMS, e-mail, tweets, documents, etc., as being related to our research. Among several drawbacks, refs. [14,32] use an expensive approach of homomorphically comparing each query word with each word of the trained model's dictionary (bag-of-words) to check for matches, resulting in high latency and computation cost. Those drawbacks were addressed by [30,31], which use secret sharing, and ref. [15] which uses a combination of homomorphic encryption and garble circuits to deal with private text classification. None of the above addressed the issue of private training. The scheme in [28] was among the rare ones to address the issue of private training over textual datasets. Although they train an ML model over a single encrypted dataset (rather than over multiple ones), the fact that they do not do any prior feature selection or pre-processing over plain (un-encrypted) data on edge makes it have high computation and communication costs. This is somehow addressed in [29], which trains an ML model over multiple datasets by offloading as much as possible of the computation on edge, in the process introducing a simple but effective feature selection by introducing bigrams of consonants for SMS-like datasets. Refs [26,29] provide relatively solid secure and private classification schemes of textual data in terms of computation and communication efficiency, as well as security and privacy characteristics.

None of the above schemes address the issues of authenticity, integrity, or protocol flow consistency, which are enabled by our proposed chain-of-blocks (COB) technology (Section 3.4). Table 1 summarizes privacy properties among several PP ML schemes.

**Table 1.** Security, and efficiency properties among different secure and private ML schemes.

| Property \ Schemes                                            | [34] | [35] | [36] | [37] | [38] | [39] | [40] | [14] | [15] | [30,31] | [32] | [26] | [27] | [28] | [29] | Ours |
|---------------------------------------------------------------|------|------|------|------|------|------|------|------|------|---------|------|------|------|------|------|------|
| Deals with secure feature selection ( <i>secFS</i> )          | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ×    | ×    | ×       | ×    | ×    | ×    | ×    | ×    | ✓    |
| Privacy of the input features for <i>secFS</i>                | ×    | ×    | ×    | ×    | ×    | ×    | ×    | NA   | NA   | NA      | NA   | NA   | NA   | NA   | NA   | ✓    |
| Privacy of input features' values for <i>secFS</i>            | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | ✓    | NA   | NA   | NA      | NA   | NA   | NA   | NA   | NA   | ✓    |
| Security and privacy of intermediate results for <i>secFS</i> | ✓×   | ✓×   | ✓×   | ✓×   | ✓×   | ✓    | ✓×   | NA   | NA   | NA      | NA   | NA   | NA   | NA   | NA   | ✓    |
| Privacy for the output (selected features) for <i>secFS</i>   | ×    | ×    | ×    | ×    | ×    | ×    | ✓×   | NA   | NA   | NA      | NA   | NA   | NA   | NA   | NA   | ✓×   |
| Deals with PP training (secure training— <i>sect</i> )        | ×    | ×    | ×    | ×    | ×    | ×    | ✓×   | ×    | ×    | ×       | ×    | ✓    | ✓    | ✓    | ✓    | ✓    |
| Privacy of the input features for <i>secT</i>                 | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA      | NA   | ×    | ×    | ×    | ×    | ✓    |
| Privacy of the input features' values for <i>secT</i>         | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA      | NA   | ✓    | ✓    | ✓    | ✓    | ✓    |
| Security and privacy of intermediate results for <i>secT</i>  | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA   | NA      | NA   | ✓    | ✓    | ✓    | ✓    | ✓    |
| Privacy of the trained model for <i>secT</i>                  | ×    | ×    | ×    | ×    | ×    | ×    | ×    | NA   | NA   | NA      | NA   | ×    | ×    | ✓    | ✓    | ✓    |

|                                                                                                                                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |
|--------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|
| Non interactive (one-round) for secure training <i>secT</i>                                                                    | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | NA | ✗  | ✓  | ✓  | ✓ | ✓ |
| Allows high (byzantine) failure rate of dataset owners during <i>secFS</i> or <i>secT</i>                                      | ✓  | ✓✗ | ✓✗ | ✓  | ✗  | ✗  | ✓  | NA | NA | NA | NA | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Has high tolerance of collisions among datasets for <i>secFS</i> or <i>secT</i>                                                | ✓✗ | ✗  | ✓✗ | ✓  | ✗  | ✗  | ✓  | NA | NA | NA | NA | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Deals with secure classification— <i>secC</i>                                                                                  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | NA | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Privacy of the trained model for <i>secC</i>                                                                                   | NA | NA | NA | NA | NA | NA | NA | ✓  | ✓  | ✓  | ✗  | ✓  | ✓✗ | ✓  | ✓  | ✓ | ✓ |
| Privacy of the user query for <i>secC</i>                                                                                      | NA | NA | NA | NA | NA | NA | NA | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Privacy of intermediate results for <i>secC</i>                                                                                | NA | NA | NA | NA | NA | NA | NA | ✓✗ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Privacy of the final classification for <i>secC</i>                                                                            | NA | NA | NA | NA | NA | NA | NA | ✗  | ✓  | ✓  | ✓  | ✓  | ✗  | ✓  | ✓  | ✓ | ✓ |
| Non interactive (one-round) for <i>secC</i>                                                                                    | NA | NA | NA | NA | NA | NA | NA | ✗  | ✓  | ✗  | ✗  | ✗  | ✗  | ✓  | ✓  | ✓ | ✓ |
| The protocol enables no loss of accuracy w.r.t to the plain classifier along the path                                          | ✓  | ✓  | ✗  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | NR | NR | ✗  | ✗  | ✓  | ✓ | ✓ |
| Resistant to the STC attack proposed in [7]                                                                                    | NA | NA | NA | NA | NA | NA | NA | ✓  | ✗  | ✗  | NA | ✗  | ✗  | ✗  | ✓  | ✓ | ✓ |
| Data Confidentiality, Integrity and Authenticity and protocol consistency (chain-of-blocks) for interactions                   | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗ | ✓ |
| Computing on plain in edge to avoid costly server homomorphic computation costs for <i>secFS</i> or <i>sect</i> or <i>secC</i> | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✗  | ✓  | ✓  | ✓  | ✓✗ | ✗  | ✗  | ✗  | ✓  | ✓ | ✓ |
| Deals with more than two EDOs                                                                                                  | ✓  | ✓  | ✓  | ✓  | ✓  | ✗  | ✗  | NA | NA | NA | NA | ✓  | ✓  | ✗  | ✓  | ✓ | ✓ |
| Fault tolerance of up to $n - 1$ out of $n$ EDO collisions                                                                     | ✓✗ | ✗  | ✗  | ✓  | ✗  | NA | ✓✗ | NA | NA | NA | NA | ✓  | ✓  | NA | ✓  | ✓ | ✓ |
| Uses a centralized server(s) to avoid communication overhead for <i>secFS</i> or <i>sect</i> or <i>secC</i>                    | ✗  | ✗  | ✗  | ✗  | ✗  | NA | ✓  | NA | NA | ✗  | ✓  | ✓  | ✓  | ✓  | NA | ✓ | ✓ |
| Avoids using multiple (more than two) semi-honest non-colluding servers for <i>secFS</i> or <i>sect</i> or <i>secC</i>         | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✗  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Classif. Cost is independent from query size                                                                                   | NA | NA | NA | NA | NA | NA | NA | ✗  | ✓  | ✗  | ✗  | NA | NA | ✗  | ✓  | ✓ | ✓ |
| Resistant to eavesdropping                                                                                                     | ✗  | ✗  | ✗  | ✗  | ✗  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Enables 2PC (i.e., 2 Dos) for <i>secFS</i> or <i>secT</i>                                                                      | ✗  | ✗  | ✗  | ✗  | ✗  | ✓  | ✓✗ | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| Applicable to the post-quantum world                                                                                           | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✓  | ✓  | ✗  | ✗  | ✓  | ✓  | ✓  | ✓  | ✓ | ✓ |
| <i>secFS</i> or <i>sect</i> cost do not dependent on the total number of records among datasets                                | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | ✗  | NA | NA | NA | NA | ✗  | ✗  | ✗  | ✓  | ✓ | ✓ |

**Legend:** ✓ = presence; ✓✗ = partial presence; ✗ = absence of property (characteristic). NA = not applicable; NR = not reported.

### 3. Participants and Adversary Models. Protocol Flows. Chain-of-Blocks

This section introduces the participants' (constituents) of the system architecture and their attack (adversary) models, provides a brief overview of the protocol flows given in this paper, and introduces the concept of chain-of-blocks.

#### 3.1. Participants (Constituents)

The proposed system architecture has the following participants (constituents):

- (1) **EDO (Edge Dataset Owner)**—We have  $n$  such EDOs in our system, denoted as  $EDO_k$ , each owns a dataset  $DS_k$ , where  $1 \leq k \leq n$ , that they are willing to use for training ML models in a secure and private fashion.
- (2) **TACS (The Aggregating Cloud Server)**: a cloud server used to do the bulk of the proposed protocols' Somewhat Homomorphic Encryption (SWHE) computation (Section 4.3).
- (3) **EDS (The Encryption Decryption Cloud Server)**: It's the only participant in the system that has a pair of public keys (whose public/private keys are  $(pk, sk)$ ) with SWHE properties (Section 4.3). All the data that are homomorphically evaluated in our protocols are encrypted using EDS' public key  $pk$ , thus it is the only one that can decrypt them with its private key  $sk$ .
- (4) **EC (Edge Client)**: has an unclassified query that he/she wishes to classify in secure and private manner by utilizing the trained ML model obtained from EDOs private datasets.

The participants together with protocol flows are illustrated in Figures 1 and 2. Protocol flows are briefly explained in Section 3.3. Their details are given in Sections 5–8.

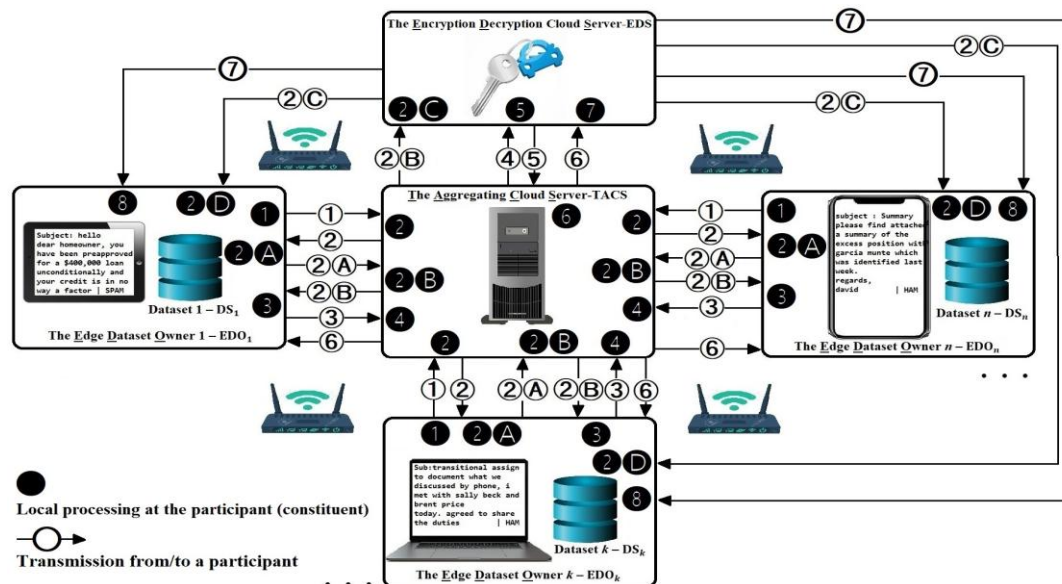


Figure 1. Protocol flows for secure feature selection (*secFS-S1* and *secF-S2*).

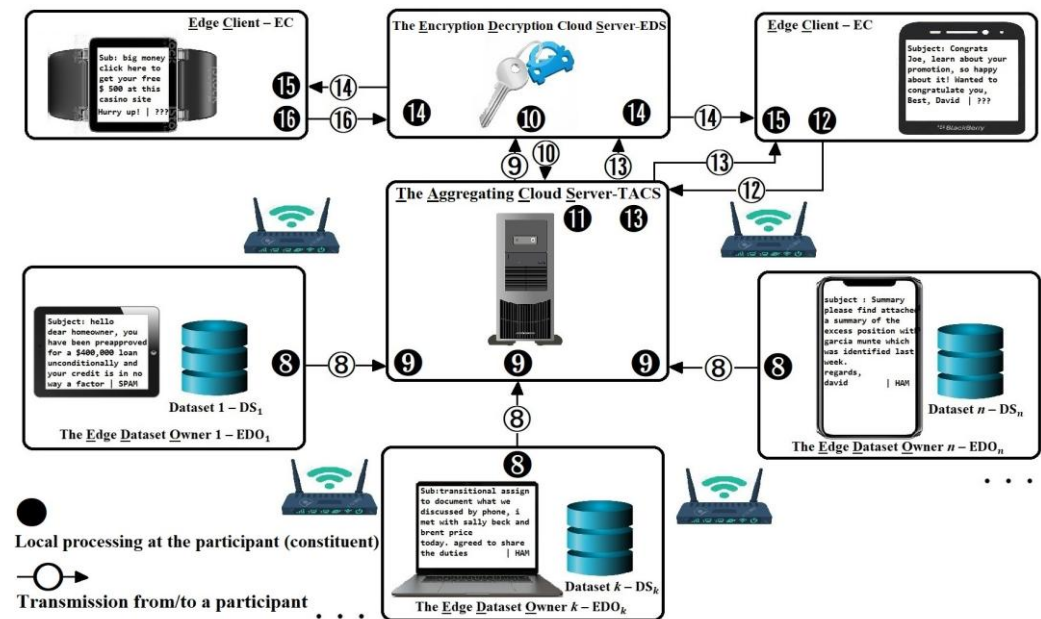


Figure 2. Protocol flows for secure training (*secT*) and secure classification (*secC*).

Each participant has a pair of public cryptosystem keys (say RSA). For EDO the pair of public/private keys is  $(pk_E, sk_E)$ ; for TACS it is  $(pk_T, sk_T)$ , for  $EDOk$  it is  $(pk_k, sk_k)$ , where  $1 \leq k \leq n$ ; and for EC it is  $(pk_{EC}, sk_{EC})$ . They form the set of keys  $KeySet = \{(pk_E, sk_E), (pk_T, sk_T), (pk_k, sk_k)_{k=1}^n, (pk_{EC}, sk_{EC})\}$ .

### 3.2. Adversary (Attack) Model of the Participants

**TACS, EDS and EC are assumed to be in the passive semi-honest (honest but curious) model**, which means that they follow the protocol but, in the background, they try to infer some private data which they are not supposed to. A formal definition of the semi-honest model is given in Section 10. We assume that TACS and EDS do not collude.

**Some EDOs are assumed to be malicious**, which means that they do not follow the protocols, thus, they input false data while participating in the protocols. The aim of malicious EDOs is to disrupt the correct usage of the whole system, which eventually is reflected in incorrect classification of unclassified client (EC) queries. Out of  $n$  EDOs, our environment setting allows for up to  $n - 3$  EDO failures, i.e. EDOs not being able to input their data while executing the protocols or simply refusing to participate in them. Moreover, the proposed protocols allow of up to  $n - 2$  collusions (cooperations) between EDOs without jeopardizing the private content of the datasets of the remaining and non-colluding EDOs (Section 10, Figures 1 and 2). Also, during the secure classification stage, we assume a more active EC that performs the malicious STC-attack proposed in [13]. The motivations for participants to behave in the manner described above are given in [11–21].

### 3.3. Protocol Flows At-a-Glance

- (1) **GRPV (Generate Random Permutation Vector)**: it is a Secure Multi-Party Computation (SMPC) protocol through which EDOs agree on a random permutation of their hash bits of their private words, needed for the secure feature selection protocol.
- (2) **secFS: (secure feature selection—stage 1 and 2)**: illustrated in Figure 1. Simultaneously, all the  $n$  EDOs find  $W_k$ , which is the set of permuted bit hashes of the local unique words (features) of their local textual datasets, where  $1 \leq k \leq n$ , ① and send them to TACS ①. TACS finds their union (the global set of hashes of unique words)  $W = \bigcup_{k=1}^n [Dec_{sk_T}(W_k)]$  ② and broadcasts it to all EDOs ②. Using each word of  $W$  as an index entry, EDOs simultaneously construct the corresponding ciphertexts of

the word counts (appearances) in ham and spam documents in the local dataset, i.e.  $hamWInDoc_{c_k}, spamWInDoc_{c_k}$ , then they construct the replicated ciphertexts of the number of ham and spam documents in the local datasets,  $hamM_{c_k}, spamM_{c_k}$ , ③ and send them to TACS ③. TACS finds the global number of ham and spam documents, as well as the global ham and spam counts for each word of  $W$  by homomorphically summing up the locally constructed EDO ciphertexts, thus  $hamWInDoc_c = \sum_{k=1}^n (hamWInDoc_{c_k})$ ,  $spamWInDoc_c = \sum_{k=1}^n (spamWInDoc_{c_k})$ ,  $hamM_c = \sum_{k=1}^n (hamM_{c_k})$  and  $spamM_c = \sum_{k=1}^n (spamM_{c_k})$ , then proceeds to find the terms needed to calculate the information gains of each word, randomizes those terms ④ and sends them to EDS ④. EDS decrypts those randomized terms, uses them to compute the vector of the randomized terms of the information gains for each word, encrypts the randomized logarithmic terms ⑤ and sends them back to TACS ⑤. TACS homomorphically removes the randomizations of the previously randomized terms and finds the information gain for each word in its corresponding index inside  $inforGains_c$ , adds some randomizations to get  $rndInforGains_c$  ⑥ and sends it to EDS while broadcasts the randomizing numbers to each EDO ⑥. EDS decrypts  $rndInforGains_c$  to get the randomized vector of information gains  $rndInforGains_v$  ⑦ and broadcasts it to the EDOs ⑦. Each of the EDOs removes the randomization from  $rndInforGains_v$  to get  $inforGains_v$  vector, which at the corresponding indexes, contains the information gain data for each of the words. EDOs then choose the top  $m$  words with the highest information gain as the selected features' set  $SF$ , thus  $SF = \{w_1, \dots, w_m\}$  ⑧. It should be noted that we choose the information gain to do the feature selection since it is shown to be among the most effective feature selecting metrics for text classification [20,34], and due to the lack of secure and private schemes that use it in literature [23–29].

- (3) **sect (secure federated learning—training)**: Illustrated in Figure 2. Using the selected words of  $SF$  as index entries,  $n$  EDOs simultaneously construct the ciphertext of the training vector which containing the local frequencies of each word,  $TV_{c_k}$ , ⑧ and send them to TACS ⑧. TACS sums them up to get the global frequencies, randomizes this result to get  $rndTV_c = (\sum_{k=1}^n TV_{c_k}) \times R_p$  ⑨ and sends it to EDS ⑨. After decrypting it, EDS finds the randomized class and conditional word-class logarithms of probabilities, integerizes and encrypts them to get the randomized trained model  $rndTM_c$  ⑩ and sends it back to TACS ⑩, which homomorphically removes the randomization to get the final trained model  $TM_c$  which represents NB or the MNB classifier ⑪.
- (4) **secC (secure ML classification)**: Shown in Figure 2. EC multiplies its unclassified query  $q_p$  with the trained model  $TM_c$  ⑫ and sends the result to TACS ⑫ for homomorphic processing and randomization ⑬. The randomized query result is sent to EDS while the randomizing data to EC ⑬. EDS decrypts the randomized query result ⑭ and sends it back to the EC ⑭, which in de-randomizes query result to get the final classification ⑮.

### 3.4. Chain-of-Blocks (COB)

All the participants have a pair of public/secret keys used for signatures and secure communications (Section 3.1). Additionally, EDS has a pair of public/secret keys with SWHE properties (Section 4.3). All these key pairs form the *KeySet* set (Sections 5 and 6).

While executing *secFS* and *secT*, each participant interacts (sends and receives data) with other participants several times! For each such interaction, each participant constructs its **log block** with the following **content** (Figure 3):

1. **Hashes of blocks of data received from other participant(s).** If data are received from multiple participants, then a Merkle Hash Tree over all input data is constructed.
2. **Hashed data sent to other participants.** If multiple data are sent to other participants, then a Merkle Hash Tree of all output data is constructed.
3. **Timestamps.**

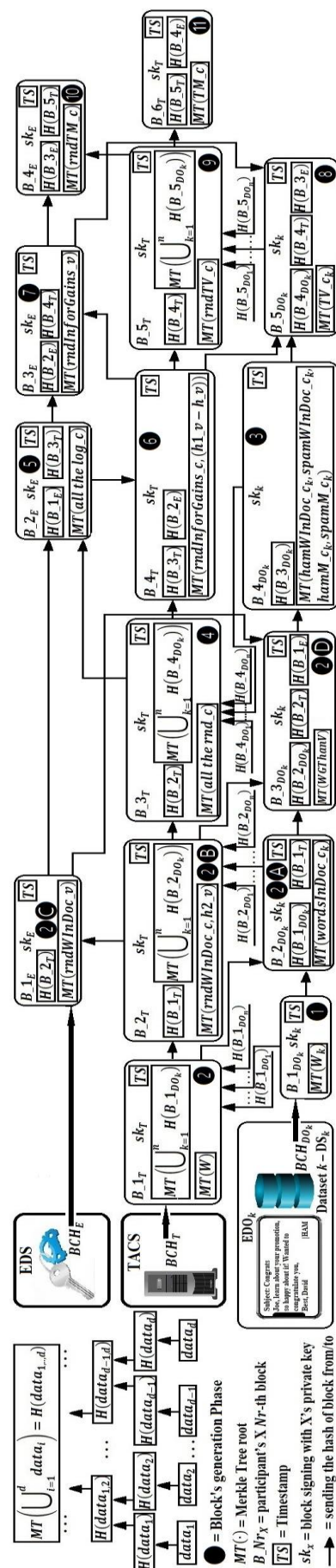
Afterwards, **each participant signs the hash of its log block** by encrypting it with his private key, **and this signature is added to what we call the chain-of-blocks (COB) (not to be confused with blockchain technology [33,42])!** Chain-of-blocks (COB) is illustrated in Figure 3 for *secFS* (Section 5) and *secT* (Section 6).

Data that participants send/receive from each other, as well as the chain-of-blocks, are kept at TACS and, together, they offer the following security and privacy properties:

1. **Confidentiality**—since all the data that participants send to each other are encrypted either by receiver's corresponding public key or the EDS's public SWHE key.
2. **Integrity**—by comparing the received and calculated hashes for each block separately. In this fashion, eventual network errors (transmission, propagation) are detected.
3. **Authenticity**—by verifying each of the signed log blocks in chain-of-blocks by using participant's corresponding public key. In this fashion, unauthorized participants are prevented from participating in the protocols.
4. **Order (flow, sequence) of execution**—by checking the timestamps in each block of the chain-of-blocks.

It should be noted that when one or more new EDOs want to contribute with their own private datasets to obtain a more accurate trained ML model, then secure feature selection (*secFS*) and secure federated learning–training (*secT*) protocols are executed from the beginning. In the process, the EDOs that participated in the previous executions are also invited to participate. For each execution of *secFS* and *secT*, their corresponding chain-of-blocks and data that participants send/receive from each other while executing the protocol are stored at TACS. After each execution of protocols, the system checks the classification accuracy of the obtained trained ML model by testing it with already correctly classified queries. If a classification accuracy lower than a certain threshold is observed, then it is assumed that it is due to **malicious EDOs**. In such a case, these malicious EDOs are flagged as malicious, their obtained trained ML model, chain-of-blocks, and other data are deleted from TACS, whereas ECs (clients) are instructed to use the last trained ML model to classify their unclassified queries.

More details about dealing with malicious EDOs and system errors (such as network transmission and propagation errors, unauthorized EDOs) are provided in Section 8.



**Figure 3.** Illustration of generating participant’s blocks in the chain-of-blocks log data structure for each iteration while executing protocols *secFS-S1*, *secFS-S2* (Section 5), and *secT* (Section 6).

#### 4. Preliminaries and Notations

Integer vectors are denoted by having “\_v” at the end of their names, e.g.,  $A_v = \{a_1, a_2, \dots, a_N\} = \{(a_i)_{i=1}^N\}$ . By  $A_v + B_v$ ;  $A_v - B_v$ ;  $A_v \times B_v$ ;  $A_v/B_v$ , we denote the component (index) wise addition, subtraction, multiplication, and division of two integer vectors, respectively.  $A_v \% \text{mod}$  denotes applying the modular operation with modulus  $\text{mod}$  to each element of  $A_v$ .  $\lceil \cdot \rceil$ ,  $\lfloor \cdot \rfloor$  denotes the ceiling and floor functions, respectively.  $H(\cdot)$  is a cryptographic hash function, and  $|S|$  denotes the cardinality of a certain set  $S$ .

$Enc_{pk_P}(\cdot)$ ,  $Dec_{sk_P}(\cdot)$  denote encryption/decryption, using public/secret keys belonging to a participant  $P$ , where  $P = \{(EDO_k)_{k=1}^n, TACS, EDS, EC\}$  (Section 3.1).

##### 4.1. Naïve Bayes (NB) and Multinomial Naïve Bayes (MNB)

Let the transactions (documents) of a certain dataset  $DS$  have a selected feature set  $SF$  consisting of  $m$  words (features); thus,  $SF = \{w_1, \dots, w_m\}$ . For simplicity and without loss of generality, let the set of classes  $C$  of  $DS$  consist of two classes, ham and spam; thus,  $C = \{c_h, c_s\}$ . Let  $f(w_i, c_j)$  denote the frequency of word  $w_i$  in documents classified (labeled) as belonging to class  $c_j$ , while  $N(w_i, c_j)$  denotes the count (number) of documents where  $w_i$  appears at least once in documents classified as belonging to  $c_j$ , where  $1 \leq i \leq m$  and  $c_j = c_h$  or  $c_j = c_s$ . If  $w_i$  appears several times in a document belonging to  $c_j$ , then its frequency  $f(w_i, c_j)$  will be incremented as many times as it appears in that particular document, while its document count  $N(w_i, c_j)$  will be incremented by one. Let  $NT$  and  $N(c_j)$  denote the total number of transactions and the number of records belonging to class  $c_j$ , respectively. Thus,  $NT = \sum_{c_j=c_h, c_j=c_s} N(c_j)$ . Let  $P(c_j) = \frac{N(c_j)}{NT}$  denote the class probabilities. Let  $P(w_i|c_j) = \frac{f(w_i, c_j)}{N(c_j)}$  and  $P(w_i|c_j) = \frac{N(w_i, c_j)}{N(c_j)}$  denote the conditional word-class probabilities for the Multinomial Naïve Bayes (MNB) and Naïve Bayes (NB) cases, respectively. Actually,  $P(c_j)$  and  $P(w_i|c_j)$ , where  $1 \leq i \leq m$  and  $c_j = c_h$  or  $c_j = c_s$ , represent the trained model  $C_{TM}(\cdot)$  for the MNB and NB classifier, respectively. To give a chance to words whose conditional probability is zero, due to their counts or frequencies being zero, we add one to each count (frequency) accompanied by adding the vocabulary size (number of unique words) to the denominator of the corresponding conditional probabilities. This process is called Laplace Smoothing. For the MNB case, un-classified queries have the form of  $q_v = \{1, f_q(w_1), \dots, f_q(w_m)\}$ , where  $f_q(w_i)$  is the frequency of the appearance of word  $w_i$  in the query  $q_v$ , while, for the NB case, it has the form of the binary vector  $q_v = \{1, N_q(w_1), \dots, N_q(w_m)\}$ , where  $N_q(w_i) = 1$  or  $0$  if  $w_i$  appears at least once in the query and  $N_q(w_i) = 0$  if  $w_i$  does not appear in the query, for  $1 \leq i \leq m$  and  $w_i \in SF$ . Using the Bayes' Theorem [29,46,47], we classify  $q_v$  as follows:

$$C_{TM}(q_v) = P(c_j|q_v) = \underset{c_j = c_h \text{ or } c_j = c_s}{\operatorname{argmax}} \frac{P(c_j)P(q_v|c_j)}{P(q_v)} \quad (1)$$

Having in mind that  $P(q_v)$  is always the same, that logarithm is a monotonically increasing function and naively assuming that appearances of words in a query are independent of each other, i.e.,  $P(q_v|c_j) = \prod_{i=1}^m P(w_i|c_j)^{f_q(w_i|c_j)}$ , then, for the NB and MNB, case (1) can be written as follows:

$$C_{TM}(q_v) = \operatorname{sign}[K(\log P(c_h) - \log P(c_s)) + \sum_{i=1}^m K f_q(w_i)(\log P(w_i|c_h) - \log P(w_i|c_s))] \quad (2)$$

, if  $C_{TM}(q_v) > 0$ , i.e.,  $\operatorname{sign} = +$ , then  $q_v$  is classified as  $c_h$  (ham), otherwise as  $c_s$  (spam).  $K$  is a constant used to integerize terms of  $C_{TM}(q_v)$  for encryption purposes (Section 4.3).

If the dataset  $DS$  is obtained by merging  $n$  other datasets denoted as  $DS_k$  for  $1 \leq k \leq n$ , then  $NT = \sum_{k=1}^n NT(k)$ ,  $N(c_j) = \sum_{k=1}^n N^{(k)}(c_j)$ ,  $f(w_i, c_j) = \sum_{k=1}^n f^{(k)}(w_i, c_j)$ , and  $N(w_i, c_j) = \sum_{k=1}^n N^{(k)}(w_i, c_j)$ , where  $NT(k)$ ,  $N^{(k)}(c_j)$ ,  $f^{(k)}(w_i, c_j)$ , and  $N^{(k)}(w_i, c_j)$  denote the number of transactions (records), number of records labeled as  $c_j$ , frequency of

word  $w_i$  in documents labeled as  $c_j$ , and the count (number) of at least one appearance of  $w_i$  in documents labeled as  $c_j$  at dataset  $k$ , respectively, for  $c_j = c_h$  or  $c_j = c_s$ .

#### 4.2. Information Gain

For the binary classification case, let  $N(\bar{w}_i, c_j) = N(c_j) - N(w_i, c_j)$  denote the count (number) of documents in the dataset labeled as belonging to class  $c_j$  where word  $w_i$  does not appear, and  $c_j = c_h$  or  $c_j = c_s$ . Let  $N(w_i) = N(w_i, c_h) + N(w_i, c_s)$  be the number of documents where word  $w_i$  appears at least once and  $N(\bar{w}) = NT - N(w_i)$  be the count (number) of documents where  $w_i$  does not appear at all. Let  $P(w_i) = \frac{N(w_i)}{NT}$  denote the probability of  $w_i$  to appear in a document,  $P(\bar{w}_i) = \frac{N(\bar{w}_i)}{NT}$  denote the probability of  $w_i$  not appearing in a document,  $P(w_i, c_j) = \frac{N(w_i, c_j)}{NT}$  denote the probability of word  $w_i$  to appear in a document classified as  $c_j$ , and  $P(\bar{w}_i, c_j) = \frac{N(\bar{w}_i, c_j)}{NT}$  denote the probability of word  $w_i$  not appearing in a document classified as  $c_j$ , where  $c_j = c_h$  or  $c_j = c_s$ . Then, the information gain (IG) of a word  $w_i$  is defined as follows [46–48]:

$$IG(w_i) = \sum_{c_j=c_h, c_s} \left( P(w_i, c_j) \log \left( \frac{P(w_i, c_j)}{P(w_i)P(c_j)} \right) + P(\bar{w}_i, c_j) \log \left( \frac{P(\bar{w}_i, c_j)}{P(\bar{w}_i)P(c_j)} \right) \right) \quad (3)$$

The information gain is a quantitative metric, which measures the reduction of entropy (uncertainty) of a query  $q_v$  to belong to a class  $c_j$  after word  $w_i$  has been observed in the query. The higher the entropy reduction, the more information gain the word  $w_i$  offers. This is the reason that makes information gain one of the most metrics in dimension reduction (feature selection), especially when choosing the top  $m$  words with the highest information gain in what is known as “bag-of-words” [46–48]. Substituting the probabilities with their counts and having in mind that we want to find the top  $m$  words with the highest IG, but not their exact IG, then (3) is rewritten as follows:

$$IG(w_i) \sim \sum_{c_j=c_h, c_s} \left( N(w_i, c_j) \log \left( \frac{N(w_i, c_j)NT}{N(w_i)N(c_j)} \right) + N(\bar{w}_i, c_j) \log \left( \frac{N(\bar{w}_i, c_j)NT}{N(\bar{w}_i)N(c_j)} \right) \right) \quad (4)$$

which makes the homomorphic evaluations and selection of the top  $m$  words with the highest IG easier (Section 4.3).

#### 4.3. SWHE Public Key Cryptosystems

Homomorphic encryption (HE) schemes allow for certain arithmetic or Boolean operations to be evaluated (done) over the ciphertexts without decrypting them (while the ciphertexts are still encrypted) [45,49–52]. So far, the strongest form of HE is Fully HE (FHE) schemes, which allow unlimited numbers of homomorphic additions and multiplications over the ciphertexts. This is due to the computationally expensive technique known as bootstrapping (homomorphic evaluation of the decryption circuit), which as an output gives a re-encrypted ciphertext over which new homomorphic operations can be done. The first FHE scheme was proposed in 2009 [49], and over the years, several others would follow [50]. All the recent FHE schemes are based on the assumption of the hardness of Decision-RLWE (Ring Learning with Errors, Section 10), **known to be resistant to quantum computer attacks** [26–29,49–52]. While the security of Somewhat HE (SWHE) schemes is also based on Decision-RLWE, they are a weaker variant of FHE in terms that they allow only a limited number of homomorphic multiplications (known as the circuit depth), but in the process, they avoid the costly bootstrapping operation [49–52]. However, in most of the real-case scenarios, the circuit depth is known in advance, which allows for the encryption parameters to be set in a way that no bootstrapping will be needed. This makes SWHE an ideal choice over FHE [14–21]. The plaintexts and the ciphertexts in FHE and SWHE schemes are polynomial rings with modulus  $X^N + 1$ , and their coefficients are integers modulo  $t$  and  $q$ , respectively, s.t.  $q \gg t$ . Thus, plaintexts belong to the ring  $R_t = \mathbb{Z}_t[X]/(X^N + 1)$  and ciphertexts to the ring  $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ . In [51], it has been

shown that if the polynomial modulus of degree  $N$  can be expressed as a multiplication of  $N$  irreducible polynomials of degree one, which in turn are automorphic to each other; then, according to the Chinese Remainder Theorem (CRT), we can encode  $N$  integers in a single plaintext or ciphertext, one integer for each polynomial coefficient. A single homomorphic operation (addition or multiplication) over two ciphertexts encoded in such a way would result in a simultaneous (parallel) component (index, slot) wise execution of the same operation over the encoded integers (Figure 4a,b)). This allows for an SIMD (Single Instruction Multiple Data) fashion of homomorphically evaluating the ciphertexts, enabling massive efficiency improvements without extra cost. Furthermore, the automorphism of the irreducible polynomials of degree one allows for the encoded integers to change their places, mainly by rotating (shifting) them to the right or left (Figure 4c)). SWHE schemes allow SIMD operations between a ciphertext and a plaintext as well, where the result is always a ciphertext. Throughout the paper, we assume that all SWHE encodings—encryptions of the plaintexts are done to support SIMD operations. For encoding/encryption purposes, plaintexts and ciphertexts are denoted by having “\_p” and “\_c” at the end of their names, respectively. Common functions of a typical public SWHE scheme are as follows [49–52]:

- $(pk, sk) = \text{KeyGen}(\lambda, N, t, q)$ . Generates a pair of public key cryptosystems (i.e., a public and corresponding secret key) according to the security parameter  $\lambda$ , polynomial modulus  $N$ , and coefficient modulus  $t$  and  $q$  for the plaintext and ciphertext, respectively.
- $m_p = \text{Encode}(m_v)$ . SIMD encoding of an integer vector into a plaintext.
- $m_c = \text{EncEncr}(m_v)$ . SIMD Encoding and encryption of an integer vector into a ciphertext.
- $C_c = A_c + B_c$ ;  $C_c = A_c + B_p$ . SIMD addition of a ciphertext with another ciphertext or plaintext (Figure 4a). The result is always a ciphertext.
- $C_c = A_c \times B_c$ ;  $C_c = A_c \times B_p$ . SIMD multiplication of a ciphertext with another ciphertext or plaintext (Figure 4b). The result is always a ciphertext.
- $B_c = \text{Rot}(A_c, R)$ . Rotating a ciphertext for  $R$  slots (indexes). If  $R > 0$ , rotations are done to the right, otherwise to the left (Figure 4c)).
- $m_v = \text{Decode}(m_p)$ . Decoding a plaintext into an integer vector.
- $m_v = \text{DecrDec}(m_c)$ . Decrypting then decoding a ciphertext into an integer vector.

Let  $A_v = \{a_1, \dots, a_N\} = \{(a_i)_{i=1}^N\}$ ,  $B_v = \{b_1, \dots, b_N\} = \{(b_i)_{i=1}^N\}$  be integer vectors and their corresponding SIMD encoded&encrypted ciphertexts  $B_c = \text{EncEncr}(A_v)$  and  $B_c = \text{EncEncr}(B_v)$ , respectively. Let  $R_v = \{R_1, \dots, R_N\} = \{(R_i)_{i=1}^N\}$  and  $h_v = \{h_1, \dots, h_N\} = \{(h_i)_{i=1}^N\}$  be random integer vectors s.t.  $R_i > 0, |h_i| < R_i$  for  $1 \leq i \leq N$ , and let  $R_p = \text{Encode}(R_v), h_p = \text{Encode}(h_v)$  be their respective SIMD encoding into plaintexts. Then  $C_c = ((A_c - B_c) \times R_p) + h_p$  is the SIMD secure comparison of the index-wise elements of  $A_v$  and  $B_v$ , firstly proposed in [29]. Namely, let  $C_v = \text{DecrDec}(C_c) = \{c_1, \dots, c_N\} = \{(c_i)_{i=1}^N\} = \{(c_i = (a_i - b_i) \times R_i + h_i)_{i=1}^N\}$ , if  $c_i = (a_i - b_i) \times R_i + h_i > 0$  then  $a_i > b_i$ , otherwise  $a_i < b_i$  for  $1 \leq i \leq N$ .

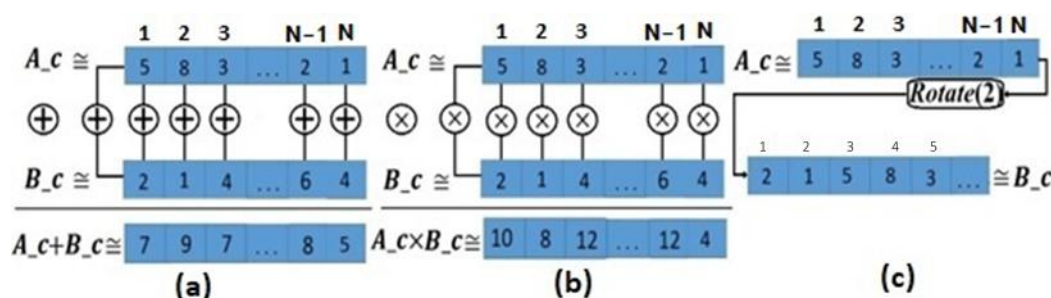


Figure 4. Illustration of SWHE SIMD (a) addition, (b) multiplication, and (c) Rotation.

## 5. Novel Secure Feature Selection Algorithm

**Definition 4.** *Feature selection* is the process of reducing the dimensions of a feature set  $F$  of a labeled dataset  $DS$  into  $SF$  according to an algorithm  $\mathcal{A}$ ; thus,  $SF = \mathcal{A}(DS, F)$ .

**GRP (Generate Random Permutation Vector):** Due to our strict security and privacy requirements (elaborated in Section 1) of keeping private both the words and their counts (frequencies), simple hashing of the words will not help. Since the EDO records (emails, documents) are in a natural spoken language (say English), by hashing all of dictionary words of that language, we can relatively easily apply the dictionary attack to find matches with the word hashes of the EDO's records. Especially, this is problematic for EDOs that wish to use a single record (mail) for training purposes (a scenario that we do not exclude), since the dictionary attack would reveal much of the record's (mail's) content. To guard against the dictionary attack, EDOs permute the hash bits of each word according to a random permutation upon which all EDOs agree. To obtain this random permutation, all the EDOs engage in a secure multi-party computation (SMPC) protocol given in Protocol 1.

**GRP:** given in Protocol 1. each of the EDOs locally constructs a random integer vector of *hashBitSize* elements, encodes and encrypts it, and sends it to TACS (lines 1–4). TACS homomorphically adds them up, adds its own random terms, send its random terms to each EDO by encrypting them with EDOs' corresponding public keys, and sends the sums of the random terms to be decrypted at EDS (lines 5–10). EDS decrypts the randomized sums and sends them to each EDO using their corresponding public keys (lines 11–13). EDOs decrypt and subtract the data obtained by EDS and TACS and apply the modulo operator with a modulus of *hashBitSize* in a component (index)-wise manner to each of them to get the random *hashBitSize* numbers placed in the *rndVec2\_v*, which are solely generated by EDOs (lines 15–16). *rndVec2\_v* is then used by the *calcPermVec()* function to generate the *hashBitPermVec\_v* by which the EDOs do the permutations of the hash bits of their words in the proceeding protocols, e.g., if in slot (index) 0 of *rndVec2\_v*, we have the value of 179, which means that the 0th bit of the original hash will be the 179th in the permuted hash bits. If in slot, say 7, of *rndVec2\_v*, we have again the value of 179, which obviously is already used in the permuted hash bits; then, we try the next slot to the right up until we find an empty place.

**secFS—Secure Feature selection (Protocol 2-Stage I and II)** is done in two stages. At the first stage, the union of all the words among  $n$  datasets are found and those that globally appear in less than *val* documents from all the datasets are filtered out, while at the second stage, the top  $m$  words with the highest IG are selected. All of this is done under strict security and privacy requirements by (1) adding the corresponding block into the chain-of-blocks, and after each interaction (Figures 2 and 3), participants securely communicate with each other either by encrypting their data with the EDS public key with SIMD SWHE properties or by using the recipients public key generated for secure communication purposes.

**secFS—S1:** Given in Protocol 2-Stage I. In Phase I, all the EDOs do some pre-processing on their local datasets, locally find all the unique words and permute their binary hashes according *hashBitPermVec\_v*, and send them to TACS (lines 1–4). In Phase II, TACS finds the global union of the permuted hashes of the words,  $W = \{\pi H(w_1), \pi H(w_2), \dots, \pi H(w_{|W|})\}$ , and sends them to all EDOs (lines 5–8). In Phase IIA, each of the EDOs, using the permuted hashes of the words in  $W$  as vector entries (indexes), locally construct  $wordsInDoc\_c_k = \{N^{(k)}(w_1), \dots, N^{(k)}(w_{|W|})\} = \left\{ \left( N^{(k)}(w_i, c_h) \right)_{i=1}^{|W|} \right\}$ , corresponding to EDO <sub>$k$</sub> 's counts of words  $w_1, \dots, w_{|W|}$  in the documents of the local dataset  $DS_k$ , where  $1 \leq k \leq n$ , and send this ciphertext to TACS (lines 9–12). In Phase IIB, TACS

encodes the replicated vector of  $N$  integers whose value is  $val$  (line 13) and then, for the needs of secure comparisons, generates and encodes random vectors  $R_v = \{R_1, \dots, R_{|W|}\} = \{(R_i)_{i=1}^N\}$  and  $h1_v = \{h1_1, \dots, h1_{|W|}\} = \{(h1_i)_{i=1}^{|W|}\}$  s.t.  $R_i > 0, |h1_i| < R_i$  for  $1 \leq i \leq |W|$  (see Section 4.3) and generates  $h2_v = \{h2_1, \dots, h2_{|W|}\} = \{(h2_i)_{i=1}^N\}$  for randomizing the secure comparison (line 13). Afterwards, it sums up all of the  $wordsInDoc_{c_k}$  to get  $wordsInDoc_c$ , which is an encryption of the vector of global counts of words appearing in documents, and performs the secure comparison proceeded with randomizations to get  $rndWInDoc_c$  (line 14). TACS sends  $rndWInDoc_c$  to EDS for decryption, while sending the randomization vector  $h2_v$  to each EDO (lines 16–18). In Phase IIC, EDS decrypts  $rndWInDoc_c$ , which looks like  $rndWInDoc_v = \left\{ \left( (N(w_i) - val) \times R_i + h1_{1,i} \right) + h2_{2,i} \right\}_{i=1}^{|W|}$  (line 19), and sends it to each EDO (lines 19–22).

Finally, in Phase IID, each EDO removes the randomization by doing the subtraction  $rndWInDoc_v - h2_v$  to get the secure comparison results  $wordsInDoc_v = \left\{ \left( (N(w_i) - val) \times R_i + h1_{1,i} \right) \right\}_{i=1}^{|W|}$  (line 24). The words  $w_i$  for which the corresponding  $\left( (N(w_i) - val) \times R_i + h1_{1,i} \right)_{i=1}^{|W|}$  term is negative (i.e. they appear in less than  $val$  global documents) are filtered out and are not part of the  $WGThanV$  set, which has  $|W'|$  words (line 25).

---

#### PROTOCOL 1: GRPV (Generate Random Permut. Vector)

---

**INPUT:**  $KeySet = \{(pk_E, sk_E), (pk_T, sk_T), (pk_k, sk_k)_{k=1}^n, (pk_{EC}, sk_{EC})\}, C, hashBitSize$

**KeySet:** the set of all the participants' public key pairs used for signing blocks in the chain-of-blocks and for secure communications.

**(pk, sk):** key pairs of EDS with SWHE properties (Section 4.3)

**hashBitSize:** the size of the hash digest in bits

---

**OUTPUT:**  $hashBitPermVec_v$

**hashBitPermVec\_v:** a vector to permuted bit hashes at EDOs

---

PHASE I—EDOs:

```

1  for  $k = 1$  to  $n$  do
2       $rndVec_v_k = generateRandomVector()$ 
3       $rndVec_c_k = EncEncr(rndVec_v_k)$ 
4      send  $rndVec_c_k$  to TACS

```

Phase II—TACS:

```

5   $rndVec_v_T = generateRandomVector()$ 
6   $rndVec_c_T = EncEncr(rndVec_v_T)$ 
7   $rndVec_c = \sum_{i=1}^k rndVec_c_k + rndVec_c_T$ 
8  send  $rndVec_c$  to EDS

```

```

9  for  $k = 1$  to  $n$  do
10     send  $Enc_{pk_k}(rndVec_v_T)$  to  $EDO_k$ 

```

Phase III—EDS:

```

11  $rndVec_v = DecrDec(rndVec_c)$ 
12 for  $k = 1$  to  $n$  do
13     send  $Enc_{pk_k}(rndVec_v)$  to  $EDO_k$ 

```

Phase IV—EDOs:

```

14 for  $k = 1$  to  $n$  do
15      $rndVec2_v = Dec_{sk_k}(rndVec_v) - Dec_{sk_k}(rndVec_v_T)$ 
16      $rndVec2_v = rndVec2_v \% hashBitSize$ 

```

17  $hashBitPermVec\_v = calcPermVec(rndVec2\_v)$

---

**PROTOCOL 2: secFS-S1 (secure Feat. Selection—Stage I)**

---

**INPUT:**  $n, \{DS_k\}_{k=1}^n, KeySet, (pk, sk), val, hashBitPermVec\_v$

$n$ : the number of EDOs

$\{DS_k\}_{k=1}^n$ : the local datasets of EDOs  $1 \leq k \leq n$

$val$ : minimum allowed global document appearance for each word

$hashBitPermVec\_v$ : a vector to permute bit hashes at EDOs, obtained by Protocol 1 (GRPV)

---

**OUTPUT:**  $WGThanV, COB$

$WGThanV$ : set of words with at least  $val$  global document appearances

---

$COB$ : chain-of-blocks (Section 3.4, Figure 3)

PHASE I—EDOs:

```

1 for  $k = 1$  to  $n$  do
2    $W_k = \pi(H(PreProcess(DS_k)), hashBitPermVec\_v)$ 
3    $B_{1_{DO_k}} = (MT(W_k), TS)$  //EDO $_k$ 's current (1st) interaction block
    $COB_{DO_k}.AddBlock(Enc_{sk_k}(B_{1_{DO_k}}))$  //added to chain-of-blocks (COB)—Figure 3
4   send  $(Enc_{pk_T}(W_k), H(B_{1_{DO_k}}))$  to TACS

```

PHASE II—TACS:

```

5  $W = \cup_{k=1}^n [Dec_{sk_T}(W_k)]$  //sorted
6  $B_{1_T} = (MT(\cup_{k=1}^n H(B_{1_{DO_k}})), MT(W), TS)$ ;  $COB_T.AddBlock(Enc_{sk_T}(B_{1_T}))$  //TACS' 1st block added to the chain-of-blocks (COB)
7 for  $k = 1$  to  $n$  do
8   send  $(Enc_{pk_k}(W), H(B_{1_T}))$  to EDO $_k$ 

```

PHASE IIA—EDOs:

```

9 for  $k = 1$  to  $n$  do
10   $wordsInDoc\_c_k = getEncryptedCounts(DS_k, Dec_{sk_k}(W))$ 
11   $B_{2_{DO_k}} = (H(B_{1_{DO_k}}), H(B_{1_T}), MT(wordsInDoc\_c_k), TS)$ ;  $COB_{DO_k}.AddBlock(Enc_{sk_k}(B_{2_{DO_k}}))$  //EDO $_k$ 's 2nd block to COB
12  send  $(wordsInDoc\_c_k, H(B_{2_{DO_k}}))$  to TACS

```

PHASE IIB—TACS

```

13  $val\_p = Enc(\{val, \dots, val\})$ ;  $(R\_v, h1\_v, h2\_v) = rndVecsforComp()$ ;  $(R\_p, h\_p) = Encode(R\_v, (h1\_v - h2\_v))$ 
14  $rndWInDoc\_c = ((\sum_{k=1}^n wordsInDoc\_c_k - val\_p) \times R\_p) + h\_p$ 
15  $B_{2_T} = (H(B_{1_T}), MT(\cup_{k=1}^n H(B_{2_{DO_k}})), MT(rndWInDoc\_c, h2\_v), TS)$ ;  $COB_T.AddBlock(Enc_{sk_T}(B_{2_T}))$  //TACS' 2nd block
16 send  $(rndWInDoc\_c, H(B_{2_T}))$  to EDS
17 for  $k = 1$  to  $n$  do
18   send  $(Enc_{pk_k}(h2\_v), H(B_{2_T}))$  to EDO $_k$ 

```

PHASE IIC—EDS:

```

19  $rndWInDoc\_v = DecrDec(rndWInDoc\_c)$ 
20  $B_{1_E} = (H(B_{2_T}), MT(rndWInDoc\_v), TS)$ ;  $COB_E.AddBlock(Enc_{sk_E}(B_{1_E}))$  //EDSs' 1st block added to the chain-of-blocks
21 for  $k = 1$  to  $n$  do
22   send  $(Enc_{pk_k}(rndWInDoc\_v), H(B_{1_E}))$  to EDO $_k$ 

```

PHASE IID—EDOs:

```

23 for  $k = 1$  to  $n$  do
24    $wordsInDoc\_v = (Dec_{sk_k}(rndWInDoc\_v) - Dec_{sk_k}(h2\_v))$ 
25    $WGThanV\_v = greaterThanVal(W, wordsInDoc\_v)$ 

```

26  $B_{3DO_k} = (H(B_{2DO_k}), H(B_{2T}), H(B_{1E}),$   
 $MT(WGThanV), TS); COB_{DO_k}.AddBlock(Enc_{sk_k}(B_{3DO_k}))$  //EDO's 3rd block added to the chain-of-blocks (COB)

**secFS-S2:** Given in Protocol 2-Stage II. It is a continuation of *secFS-S1*. In Phase III, using each of the  $|W'|$  words in *WGThanV* as index entries, each EDO locally constructs vectors  $hamWInDoc_v_k = \{(N^{(k)}(w_i, c_h))_{i=1}^{|W'|}\}$  and  $spamWInDoc_v_k = \{(N^{(k)}(w_i, c_s))_{i=1}^{|W'|}\}$ , as well as the replicated  $hamM_v_k = \{N^{(k)}(c_h), \dots, N^{(k)}(c_h)\}$  and  $spamM_v_k = \{N^{(k)}(c_s), \dots, N^{(k)}(c_s)\}$ , containing the number of local ham and spam mails (documents), respectively, replicated for  $|W'|$  times. After encoding and encrypting all of them, they are sent to TACS (lines 1–4). In Phase IV, TACS homomorphically aggregates (sums up) those vectors to get the global counts for each word to appear in ham and spam documents, as well as the global number of ham and spam mails (lines 7–10). Then, it finds  $nrMails_c$ ,  $wordsInDoc_c$ ,  $wordsNOTInDoc_c$ ,  $hamWNOTInDoc_c$ , and  $spamWNOTInDoc_c$  (lines 12–15, Figure 5), which in turn are used to find all of the randomized nominator and denominator terms inside the logarithms in (4) for each word (lines 16–23), denoted as  $allRndNomDenom_c$ , and all of them are sent to EDS (line 25), e.g., in Figure 6a), we illustrate the SIMD evaluations and randomizations of the nominator and denominator of the term  $\frac{N(w_i, c_j)NT}{N(w_i)N(c_j)}$  in (4) done in lines 16 and 20. The nominators and denominators of the other terms in (4) are found in a similar way. In Phase V, EDS decrypts all of the  $allRndNomDenom_c$  sent by TACS (lines 27) and finds the encryption of all of the logarithmic terms of (4) in randomized form, denoted as  $allTheRndLogs_c$ , e.g.,  $logHamWInDoc_c = EncEncr(K1 \times \log(rndHamWInD\_mul\_nrM\_v / rdWInDoc\_mul\_hamM\_v)) = EncEncr\left(K1 \cdot \log\left(\frac{N(w_i, c_h) \cdot NT \cdot R_{1,i}}{N(w_i) \cdot N(c_h) \cdot R_{5,1}}\right)_{i=1}^{|W'|}\right)$  (upper vector in Figure 6b), where  $K1$  is a constant used for integerization purposes and  $R_{1,i}, R_{5,i}$  are random numbers (lines 28–31). Afterwards, EDS sends the  $allTheRndLogs_c$  to TACS (line 34). In Phase VI, TACS removes the randomizations and multiplies each of the  $allTheRndLogs_c$  with the corresponding term of (4) (lines 35–42), e.g.,  $tmp\_c_1 = hamWInDoc_c \times (logHamWInDoc_c + invLogHamWInD\_p)$  in line 39 is illustrated in Figure 6b, corresponds to term  $N(w_i, c_j) \log\left(\frac{N(w_i, c_j)NT}{N(w_i)N(c_j)}\right)$  of (4). The other logarithmic terms multiplied with the corresponding counts of (4) are found in a similar way. After homomorphically finding and randomizing the information gains to get  $rndInforGains_c$  (lines 43–46), it is sent to EDS for decryption, while a partial portion of the randomization is sent to each EDO (lines 48–50). In Phase VII, EDS decrypts  $rndInforGains_c$  and sends it to each EDO. Finally, in Phase VIII, each of the EDOs partially removes the randomization to get  $inforGains_v = \{(IG(w_i) \times R + h)_{i=1}^{|W'|}\}$  (line 58). Since all the  $IG(w_i)$  terms are multiplied and added to the same random  $R$  and  $h$ , respectively, it is easy for each of the EDOs to find the top  $m$  words with the highest IG. Afterwards, EDOs apply the inverse permutation to word hashes using  $hashBitPermVec_v$  proceeded with sorting w.r.t. to hash values, thus getting the selected features set,  $SF$  (line 59). Sorting is done for performance reasons when users prepare their queries according to the selected  $m$  words during *secC*.

---

#### PROTOCOL 2: *secFS-S2* (secure Feature Selection-Stage II)

---

INPUT:  $n, \{DS_k\}_{k=1}^n, KeySet, (pk, sk), m, WGThanV$

$m$ : the number of features (words) to be selected at the end

*WGThanV*: words with at least *val* global document appearances

---

OUTPUT:  $SF = \{H(w_1), \dots, H(w_m)\}, COB$

**SF:**  $m$  selected words with the highest IG sorted by their hashes

#### PHASE III—EDOs

```

1 for  $k = 1$  to  $n$  do
2    $(hamWinDoc\_c_k, spamWinDoc\_c_k, hamM\_c_k, spamM\_c_k) = getEncryptedCounts(DS_k, wordsGThanV\_v)$ 
3    $B_{4DO_k} = (H(B_{3DO_k}), MT(hamWinDoc\_c_k, spamWinDoc\_c_k, hamM\_c_k, spamM\_c_k), TS)$  //EDO's 4th block
4   send  $(hamWinDoc\_c_k, spamWinDoc\_c_k, hamM\_c_k, spamM\_c_k, H(B_{4DO_k}))$  to TACS
5    $COB_{DO_k}.AddBlock(Enc_{sk_k}(B_{4DO_k}))$  //added to chain-of-blocks-Figure 3

```

#### PHASE IV-TACS

```

6    $(R1\_p, \dots, R8\_p) = RndEncodedVectors()$ 
7    $hamWinDoc\_c = \sum_{k=1}^n (hamWinDoc\_c_k)$ 
8    $hamM\_c = \sum_{k=1}^n (hamM\_c_k)$ 
9    $spamWinDoc\_c = \sum_{k=1}^n (spamWinDoc\_c_k)$ 
10   $spamM\_c = \sum_{k=1}^n (spamM\_c_k)$ 
11   $nrMails\_c = hamM\_c + spamM\_c$ 
12   $wordsInDoc\_c = hamWinDoc\_c + spamWinDoc\_c$ 
13   $wordsNOTInDoc\_c = nrMails\_c - wordsInDoc\_c$ 
14   $hamWNOTInDoc\_c = hamM\_c - hamWinDoc\_c$ 
15   $spamWNOTInDoc\_c = spamM\_c - spamWinDoc\_c$ 
16   $rndHamWinD\_mul\_nrM\_c = (hamWinDoc\_c \times nrMails\_c) \times R1\_p$ 
17   $rndHamWNOTInD\_mul\_nrM\_c = (hamWNOTInDoc\_c \times nrMails\_c) \times R2\_p$ 
18   $rndSpamWinD\_mul\_nrM\_c = (spamWinDoc\_c \times nrMails\_c) \times R3\_p$ 
19   $rndSpamWNOTInD\_mul\_nrM\_c = (spamWNOTInDoc\_c \times nrMails\_c) \times R4\_p$ 
20   $rndWinDoc\_mul\_hamM\_c = (wordsInDoc\_c \times hamM\_c) \times R5\_p$ 
21   $rndWNOTInDoc\_mul\_hamM\_c = (wordsNOTInDoc\_c \times hamM\_c) \times R6\_p$ 
22   $rndWinDoc\_mul\_spamM\_c = (wordsInDoc\_c \times spamM\_c) \times R7\_p$ 
23   $rndWNOTInDoc\_mul\_spamM\_c = (wordsNOTInDoc\_c \times spamM\_c) \times R8\_p$ 
24   $B_{3T} = (H(B_{2T}), MT(\bigcup_{k=1}^n H(B_{4DO_k}), MT(allRndNomDenom\_c), TS))$  //TACS' 3rd block
25  send  $(allRndNomDenom\_c, H(B_{3T}))$  to EDS
26   $COB_T.AddBlock(Enc_{sk_T}(B_{3T}))$  //added to chain-of-blocks—Figure 3

```

#### PHASE V—EDS

```

27   $(rndHamWinD\_mul\_nrM\_v, rndHamWNOTInD\_mul\_nrM\_v, rndSpamWNOTInD\_mul\_nrM\_v, rndSpamWNOTInD\_mul\_nrM\_v,$ 
 $rndWinDoc\_mul\_hamM\_v, rndWNOTInDoc\_mul\_hamM\_v, rndWinDoc\_mul\_spamM\_v, rndWNOTInDoc\_mul\_spamM\_v) =$ 
 $DecrDec(allRndNomDenom\_c \text{ sent by TACS})$ 
28   $logHamWinDoc\_c = EncEncr(K1 \times \log(rndHamWinD\_mul\_nrM\_v/rndWinDoc\_mul\_hamM\_v))$ 
29   $logHamWNOTInDoc\_c = EncEncr(K1 \times \log(rndHamWNOTInD\_mul\_nrM\_v/rndWNOTInDoc\_mul\_hamM\_v))$ 
30   $logSpamWinDoc\_c = EncEncr(K1 \times \log(rndSpamWinD\_mul\_nrM\_v/rndWinDoc\_mul\_spamM\_v))$ 
31   $logSpamWNOTInDoc\_c = EncEncr(K1 \times \log(rndSpamWNOTInD\_mul\_nrM\_v/rndWNOTInDoc\_mul\_spamM\_v))$ 
32   $B_{2E} = (H(B_{1E}), H(B_{3T}), MT(allTheRndLogs\_c), TS)$ 
33   $COB_E.AddBlock(Enc_{sk_E}(B_{2E}))$  //EDS'2nd block in chain-of-blocks (COB)—Figure 3
34  send  $(allTheRndLogs\_c, H(B_{2E}))$  to TACS

```

#### PHASE VI—TACS

```

35   $invLogHamWinD\_p = Encode(K1 \times \log(R5\_v/R1\_v))$ 
36   $invLogHamWNOTInD\_p = Encode(K1 \times \log(R6\_v/R2\_v))$ 
37   $invLogSpamWinD\_p = Encode(K1 \times \log(R7\_v/R3\_v))$ 

```

---

```

38  $invLogSpamWNOTInD_p = Encode(K1 \times \log(R8_v/R4_v))$ 
39  $tmp_{c_1} = hamWInDoc_c \times (\log HamWInDoc_c + invLogHamWInD_p)$ 
40  $tmp_{c_2} = hamWNOTInDoc_c \times (\log HamWNOTInDoc_c + invLogHamWNOTInD_p)$ 
41  $tmp_{c_3} = spamWInDoc_c \times (\log SpamWInDoc_c + invLogSpamWInD_p)$ 
42  $tmp_{c_4} = spamWNOTInDoc_c \times (\log SpamWNOTInDoc_c + invLogSpamWNOTInD_p)$ 
43  $inforGains_c = \sum_{i=1}^4 tmp_{c_i}$ 
44  $R_v = \{R, \dots, R\}; h1_v = \{h_{1,1}, \dots, h_{N,1}\}; h2_v = \{h, \dots, h\}$ 
45  $(R_p, h1_p) = Encode(R_v, h1_v)$ 
46  $rndInforGains_c = (inforGains_c \times R_p) + h1_p$ 
47  $B_{4_T} = (H(B_{3_T}), H(B_{2_E}), MT(rndInforGains_c, (h1_v - h_v)), TS)$  //TACS' 4th chain-of-blocks (COB) block
48  $send(rndInforGains_c, H(B_{4_T}))$  to EDS
49 for  $k = 1$  to  $n$  do
50    $\rightarrow send(Enc_{pk_k}(h1_v - h_v), H(B_{4_T}))$  to  $EDO_k$ 
51  $COB_T.AddBlock(Enc_{sk_T}(B_{4_T}))$  //add TACS' 4th block to chain-of-blocks (COB)
PHASE VII-EDS:
52  $rndInforGains_v = DecrDec(rndInforGains_c)$ 
53  $B_{3_E} = (H(B_{2_E}), H(B_{4_T}), MT(rndInforGains_v), TS)$ 
54 for  $k = 1$  to  $n$  do
55    $\rightarrow send(Enc_{pk_k}(rndInforGains_v), H(B_{3_E}))$  to  $EDO_k$ 
56  $COB_E.AddBlock(Enc_{sk_E}(B_{3_E}))$  //EDS' 3rd block added to the chain-of-blocks (COB)—Figure 3
PHASE VIII-EDOs
57  $rndInforGains_v = Dec_{sk_k}(rndInforGains_v)$ 
58  $inforGains_v = rndInforGains_v - Dec_{sk_k}(h1_v - h_v)$ 
59  $SF = sort(\pi^{-1}(topM(inforGains_v)), hashBitPermVec_v)$ 

```

---

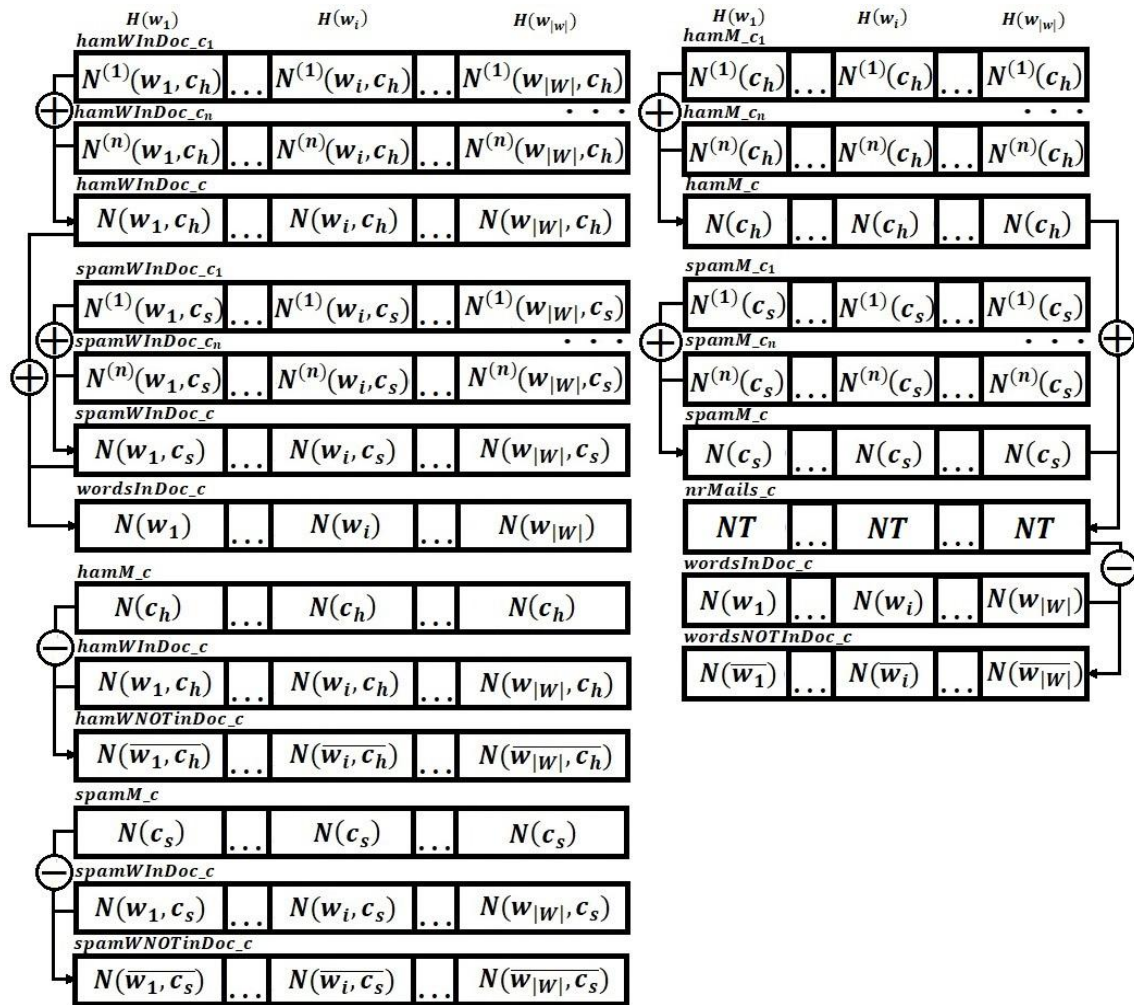


Figure 5. Illustration of SIMD evaluation of values in lines 13–16 of *secFS-S2*.

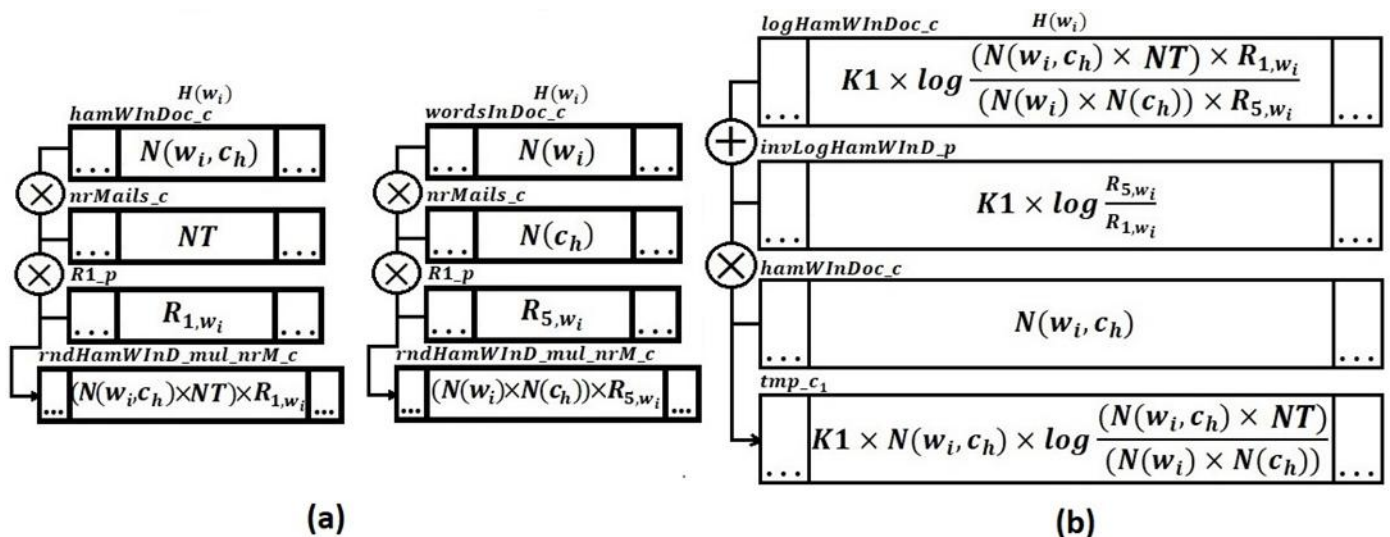


Figure 6. Illustration of SWHE SIMD evaluations of terms in (4) in *secFS-S2*.

## 6. Secure ML Federated Learning (Training)

**Definition 5.** *ML model training* is the process of acquiring parameters of the trained model  $TM$  from a labeled dataset(s)  $DS$  consisted of a selected feature set  $SF$  according to an algorithm  $T$ ; thus,  $TM = T(DS, SF)$ .

**secT—secure (Federated Learning) Training.** Given in Protocol 3. In Phase VIII, each of the EDOs locally constructs the training vector,  $TV_{v_k}$ , s.t. the first and the  $(m+1)$ -th index have the local ham  $N^{(k)}(c_h)$  and spam  $N^{(k)}(c_s)$  counts, respectively, while the indexes from 1 to  $m$  contain the local ham frequencies  $f^{(k)}(w_i, c_h)$ , and indexes from  $m+2$  until  $2m+2$  have the local spam frequencies  $f^{(k)}(w_i, c_s)$ , corresponding to the words of the selected features' set  $SF$ , respectively, where  $1 \leq i \leq m$  and  $1 \leq k \leq n$  (the upper vectors of Figure 7, lines 2–3, 5). For the needs of Laplace Smoothing (Section 4.1), an arbitrarily chosen EDO adds an extra 1 to the indexes corresponding to word frequencies and adds the dictionary size (in our case  $m$ ) to the first and  $(m+1)$ -th index corresponding to ham and spam counts. After encoding and encrypting  $TV_{v_k}$  to get  $TV_{c_k}$ , they are sent to TACS, which, in Phase IX, homomorphically sums them up and randomizes this sum to get  $rndTV_c$  (line 8, Figure 7) and sends it to EDS (line 12). In Phase X, EDS finds the randomized logarithmic terms of the MNB trained model (Section 4.1), encrypts them to get  $rndTM_c$  (Figure 8, lines 14–16), and sends it to TACS (line 18). Finally, in Phase XI, TACS removes the randomizations and subtracts the rotated result (Figure 8, lines 20–21) to get the final MNB trained model according to (2). In *secT* we continue the chain-of-blocks started in the previous protocol. Note: if instead of the frequencies  $f^{(k)}(w_i, c_h)$  and  $f^{(k)}(w_i, c_s)$ , we put  $N^{(k)}(w_i, c_h)$  and  $N^{(k)}(w_i, c_s)$  at  $TV_{v_k}$ s in Phase I, the final trained model will be the one based on NB (Section 4.1).

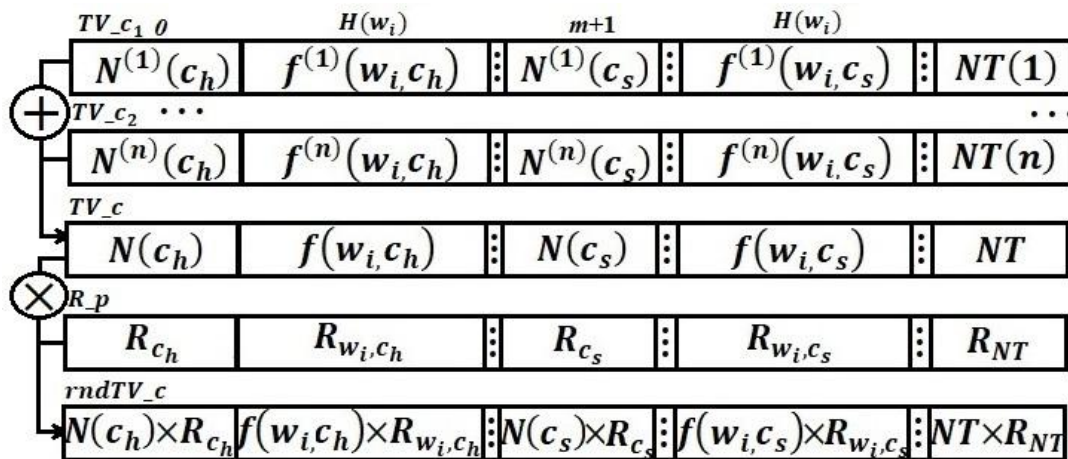


Figure 7. Obtaining and randomizing the global frequencies in SIMD fashion.

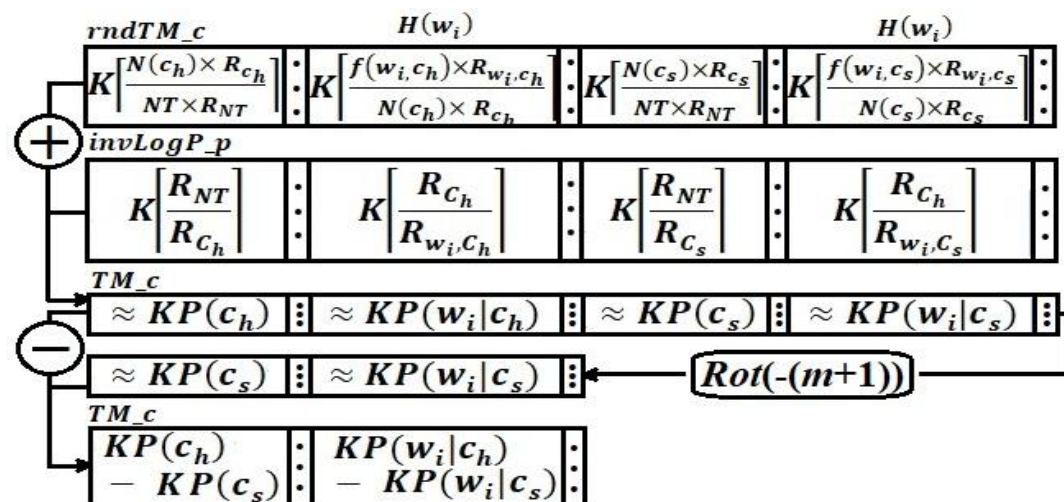


Figure 8. De-randomizing and rotating  $rndTM_c$  to get the final trained model  $TM_c$ .

**PROTOCOL 3: *secT* (secure Training)**

**INPUT:**  $n, \{DS_k\}_{k=1}^n, KeySet, SF, m$

$SF = \{H(w_1), \dots, H(w_m)\}$ :  $m$  selected features with the highest IG

**OUTPUT:**  $TM\_c = \{K(P(c_h) - P(c_s)), K(P(c_h|w_1) - P(c_s|w_1)), \dots, K(P(c_h|w_m) - P(c_s|w_m))\}$

$TM\_c$ : the (packed) trained model ciphertext

PHASE VIII—EDOs:

```

1 for  $k = 1$  to  $n$  do
2    $TV\_v_k = getTVAccordingToSF(DS_k, SF)$ 
3    $TV\_c_k = EncEncr(TV\_v_k)$ 
4    $B_{5DO_k} = (H(B_{4DO_k}), H(B_{4T}), H(B_{3E}), MT(TV\_c_k), TS)$ 
5   send  $(TV\_c_k, H(B_{5DO_k}))$  to TACS
6    $COB_{DO_k}.AddBlock(Enc_{sk_k}(B_{5DO_k}))$  //add EDO's 5th block to the chain-of-blocks (COB)-Figure 3

```

PHASE IX—TACS:

```

7  $R\_v = \{R_{C_h}, \dots, R_{w_i, C_h}, \dots, R_{C_s}, \dots, R_{w_i, C_s}, \dots, R_{NT}\}$ 
8  $rndTV\_c = (\sum_{k=1}^n TV\_c_k) \times R\_p$ 
9  $invLogP\_v = K \times \left\{ \left\lceil \frac{R_{NT}}{R_{C_h}} \right\rceil, \dots, \left\lceil \frac{R_{C_h}}{R_{w_i, C_h}} \right\rceil, \dots, \left\lceil \frac{R_{NT}}{R_{C_s}} \right\rceil, \dots, \left\lceil \frac{R_{C_h}}{R_{w_i, C_s}} \right\rceil \right\}$ 
10  $invLogP\_p = Encode(invLogP\_v)$ 
11  $B_{5T} = (H(B_{4T}), MT(\cup_{k=1}^n H(B_{5DO_k})), MT(rndTV\_c), TS)$ 
12 send  $(rndTV\_c, H(B_{5T}))$  to EDS
13  $COB_T.AddBlock(Enc_{sk_T}(B_{5T}))$  //add TACS' 5th block to the chain-of-blocks (COB)-Figure 3

```

PHASE X—EDS

```

14  $rndTV\_v = DecrDec(rndTV\_c)$ 
15  $rndTM\_v = calculateRndLogsOfProbs(rndTV\_v)$ 
16  $rndTM\_c = EncEncr(rndTM\_v)$ 
17  $B_{4E} = (H(B_{3E}), H(B_{5T}), MT(rndTM\_c), TS)$ 
18 send  $(rndTM\_c, H(B_{4E}))$  to TACS
19  $COB_E.AddBlock(Enc_{sk_E}(B_{4E}))$  //add EDS' 4th block to the chain-of-block (COB)-Figure 3

```

PHASE XI—TACS:

```

20  $TM\_c = rndTM\_c + invLogP\_p$ 
21  $TM\_c = TM\_c - Rot(TM\_c, -(m+1))$ 
22  $B_{6T} = (H(B_{5T}), H(B_{4E}), MT(TM\_c), TS)$ 
22  $COB_T.AddBlock(Enc_{sk_T}(B_{6T}))$  //add TACS' 6th block to the chain-of-block (COB)-Figure 3
23 return  $TM\_c$ 

```

**Improvement 1:** after obtaining it, the trained model  $TM\_c$  is rarely changed in practice. We can utilize this fact to send it only once to the EC and amortize the communication cost among all the secure classification instances, which will be initiated by the EC.

## 7. Secure ML Classifications

**Definition 6. Classification** is the process of assigning a label to an unlabeled query  $q$  according to a trained model  $TM$  using an algorithm  $\mathcal{C}$ ; thus,  $C_{TM}(q) = \mathcal{C}(TM, q)$ .

**secC—secure classification.** Given in Protocol 4. In Phase XII, EC encodes its query  $q_v$ , which has 1 in the first slot (index), proceeded by the query frequencies of each word in the query ordered according to the word arrangement in the selected features sets  $SF$  (line 1, Figure 9). After multiplying the encoded query with the trained model  $TM_c$ , it sends the result to TACS (lines 2–3, Figure 9). In Phase XIII, in accordance with (2), TACS homomorphically finds the sums of each of the  $(m + 1)$  slots in  $\log_2(m + 1)$  rotations and additions with the result residing in the first slot (lines 5–6). To do this,  $(m + 1)$  should be a power of two. If it is not the case, then we can pad extra slots with dummy values (usually zeros). Then, for secure comparison purposes (Section 4.3), it multiplies this result with a random  $R_p$  followed by adding  $h_p$  s.t.  $R_p$ , and  $h_p$  are constructed having in mind the secure comparison requirements for them in Section 4.3. To protect the result from the STC (substitute-then-compare) attack from [13], TACS adds an extra random  $h2_p = Encode(h_v)$  to the result and sends the randomized result to EDS and the  $h2_v$  to EC (lines 4–11). In Phase XIV, EDS decrypts the randomized classification result and sends it to EC (lines 12–13). Finally, in Phase XV, EC subtracts  $h2_v$  from the randomized classification result and, according to (2), gets the final classification based on the sign of the result (lines 14–18). If the query  $q_v$  instead of the frequencies contains the counts ( $N_q(w_i)$ ) of words appearing in the query, then instead of MNB, we are dealing with the NB.

---

**PROTOCOL 4: secC (secure Classification)**


---

**INPUT:**  $SF, TM_c, q_v = \{1, f_q(w_1), \dots, f_q(w_m)\}$ :

$(pk, sk)$ : key pairs of EDS with SWHE properties

$SF$ : the set of  $m$  selected features with the highest IG, selected (filtered) during the *secFS* protocols

$TM_c$ : the trained model ciphertext residing at EC

$q_v$ : the EC's (packed) query vector(s)

---

**OUTPUT:**  $C_{TM}(q)$

$C_{TM}(q)$ : the  $q_v$ 's final classification

---

PHASE XII—EC:

1  $q_p = Encode(q_v)$

2  $C_{TM}(q)_c = TM_c \times q_p$

3 send  $C_{TM}(q)_c$  to TACS

PHASE XIII-TACS

4 for  $i = 0$  to  $\lceil \log(m + 1) \rceil$

5  $C_{TM}(q)_c = C_{TM}(q)_c + Rot(C_{TM}(q)_c, -2^i)$

6  $(R_p, h_p, h2_p) = EncoderndVecsforComp(R_v, h_v, h2_v)$

7  $rndC_{TM}(q)_c = ((C_{TM}(q)_c \times R_p) + h_p) + h2_p$  //to protect from the STC attack described in [7]

8 send  $rndC_{TM}(q)_c$  to EDS

9 send  $Enc_{pk_{EC}}(h2_v)$  to EC

PHASE XIV-EDS

10  $rndC_{TM}(q)_v = DecrDec(rndC_{TM}(q)_c)$

11 send  $Enc_{pk_{EC}}(rndFinClas_v)$  to EC

PHASE XV-EC

12  $res_v = Dec_{sk_{EC}}(rndC_{TM}(q)_v) - Dec_{sk_{EC}}(h2_v)$

13 if  $(res_v \geq 0)$  return  $C_{TM}(q) = \text{"ham"}$

14 else return  $C_{TM}(q) = \text{"spam"}$

---

**Improvement 2:** We can convert *secC* from a 3PC to 2PC by executing Phases XII and XIII at EC (will denote them as Phase XVI), and we can proceed with Phase IV at EDS and Phase XV at EC as usual (the EC on the left side in Figure 2).

**Improvement 3:** For high throughput, we can simultaneously process up to  $q = \lfloor \frac{N}{(m+1)} \rfloor$  queries by obtaining a replicated  $TM\_c$  during *secT* (which can be done without extra costs, Figure 8) and encoding (packing)  $q$  queries to  $q\_p$  during *secC*.

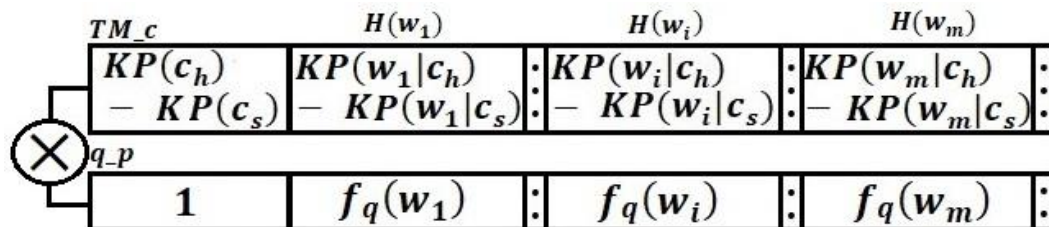


Figure 9. SWHE SIMD multiplication of the trained model  $TM\_c$  with query  $q\_p$ .

## 8. Dealing with Malicious EDOs and System Errors

The algorithm is given in Protocol 5. It deals with system errors, such as detecting network (transmission or propagation) errors and unauthorized participants, as well as dealing with malicious EDOs, which deviate from the protocol by providing false data.

In line 1, function *checkIntegrity(...)* uses *AllTransData* to reconstruct Chain-of-Blocks (COB) (Section 3.4) and compares its calculated hashes with the hash values at the original COB. It returns true if hashes match, i.e., the integrity check holds; otherwise, it returns false, which is probably due to network transmission or propagation errors.

In line 2, function *checkSignatureValidity(...)* uses the set of public keys of participants *KeySet* to check and verify the signatures for each block of each participant at COB (Figure 3). It returns true if all signature verifications hold. Otherwise, it returns false, which signals that one or more unauthorized participants participated in protocols 1–3.

In line 3, function *checkOrderOfExecution(...)* uses timestamps at each COB block to check the order (sequence, consistency) of participants participating and interacting with each other in protocols. It returns true if the orders holds; otherwise, it returns false.

In lines 4–8, using the trained ML model  $TM\_c$ , the protocol classifies an array of  $nrQ$  already correctly classified queries and compares the obtained classification labels with the correct ones (lines 7–8). Afterwards, it uses this information to calculate the classification accuracy of  $TM\_c$  (line 9). If the calculated accuracy is smaller than a threshold *thresh* (line 10), then it is assumed that it is due to the presence of malicious EDOs. In such a case, EDOs that participated in Protocols 1–3, to obtain the current  $TM\_c$ , are added to a list of potentially malicious EDOs (line 11).

If any of the integrity (line 1), or authorization (line 2) or consistency (line 3) or malicious EDO checks (line 12) fails, then the current  $TM\_c$ , *SF* and *COB* are deleted from the system, and the old ones (from the previous run of Protocols 1–3) are rolled (brought) back to be used to classify unclassified client (EC) queries.

---

### PROTOCOL 5: *MalEDO\_SE* (Dealing with Malicious EDOs (Edge Dataset Owners) and System Errors)

---

INPUT: *COB*, *AllTransData*, *KeySet*,  $TM\_c$ , *SF*, *testQueries*[ $nrQ$ ], *trueLabels*[ $nrQ$ ], *thresh*

---

**COB:** the chain-of-Blocks (COB) (Section 3.4) obtained during *secFS* and *secT* protocols (Sections 5 and 6) (Figure 3)

**AllTransData:** all data that participants send to each other while running *secFS* and *secT*, needed to construct COB (Figure 3)

**KeySet:** the set of all the participants' public key pairs used for signing blocks in the chain-of-block and for secure communications.

**$TM\_c$ :** the trained model ciphertext (which is the output of *secT*) residing at TACS

**SF:** the set of  $m$  selected features with the highest IG, selected (filtered) during the *secFS* protocols

**testQueries[nrQ]**: an array of *nrQ* correctly labeled (classified) queries used to test the accuracy of *TM\_c*

**trueLabels[nrQ]**: the correct corresponding labels (classes) of the *nrQ* queries of the array *testQueries[nrQ]*

**thresh**: the minimum accepted ML classification accuracy threshold value by the system

**OUTPUT: *decision***

**decision**: *accept*, if no system errors and ML classification accuracy is above threshold *thresh*, otherwise *decline*

```

1 integrityCheck = checkIntegrity(COB, AllTransData) //returns true if integrity check holds, otherwise returns false
2 authorizationCheck = checkSignatureValidity(COB, KeySet) //true if authorization (identity) check holds, otherwise false
3 consistencyCheck = checkOrderOfExectuion(COB) //true if order (sequence) of execution holds, otherwise false
4 correctlyClassifiedQueeries = 0; noMaliciousEDO = true
5 for i = 0 to nrQ
6     obtainedLabel[i] = secC(SF, TM_c, testQueries[i])
7     if (trueLabels[i] == obtainedLabel[i])
8         correctlyClassifiedQueeries ++
9 accuracy =  $\frac{\text{correctlyClassifiedQueeries}}{\text{nrQ}}$ 
10 if (accuracy < thresh)
11     addToSuspectedMaliciusListParticipatingEDOs()
12     maliciousEDOCheck = false
13 if (integrityCheck == false or authorizationCheck == false or consistencyCheck == false or noMaliciousEDO = true)
14     rollBackToLastTrainedModel()
15     return decline
16 otherwise return accept

```

## 9. Experimental Evaluations and Comparisons

We run our codes in a machine with an Intel Core i3-4000M processor with two physical cores, each of 2.4 GHz (we utilize one core only in our implementations) with 4 GB of DDR3 RAM and 64-bit Windows 10 Pro as an OS. For SWHE purposes, we use the C++-based Microsoft's SEAL 3.2 library [53], which implements the SWHE scheme in [52].

For benchmark and comparison purposes with the state-of-the-art schemes, we used two textual datasets, the SMS spam corpus [54] and the Enron e-mail dataset [47,55]. Their implementations can be found in [55,56], respectively, for a total of around 15 000 lines of original C++ code. The Enron e-mail dataset is a collection of e-mails from 150 user profiles, with a total of 16,555 ham and 17,148 spam e-mails. For the dataset pre-processing in line 2 of the *secFS – S1* (Protocol 2), we applied stop-words removal, partial punctuation removal, stemming by using the library in [57] and converting all the letters to lowercase. For our protocols, the input parameter values for the Enron dataset are  $val = 5$ ,  $K_1 = 100\,000$ ,  $K = 256$ , and  $m = 2047$ . For the SMS spam corpus, which has 4591 ham and 983 spam SMSes, instead of *secFS – S1* and *secFS – S2* (Protocols 2 and 3), we use the SMS consonant bigram approach proposed in [29] accompanied by several improvements in training and classification. This gives us a total of  $m = 21 \times 21 = 441$  features for the SMSes vs. thousands for the classical “bag-of-words” approach. Furthermore, in the trained model *TM\_c*, instead of allocating separate slots for each ham- and spam-related probability as it is done in [29], we allocate a single slot, which has the subtractions of the corresponding ham and spam probabilities (Sections 6 and 7, Figures 8 and 9). This alone gives an improvement of 50% in terms of computation and communication costs with respect to the private SMS algorithms in [29]. For both the datasets, unlike the related schemes, we also applied Laplace smoothing to boost our classification accuracy (Sections

4.1 and 6). Compared with the reported accuracy results in related research, such as [15,28] and [47], our results are shown to be among the most accurate ones in the literature in both private and non-private ML classifications. Also, due to choosing  $K$  to be between 8 and 10 bits, there was no loss of accuracy due to private classifications, which is consistent with the findings in [29].

For the SWHE parameters (Section 4.3) for  $N = 8192, 16,384$ , we use the corresponding values of  $t = 37, 60$  bits and  $q = 218, 438$  bits, respectively, and security of  $\lambda = 128$  bits.

Table 2 gives the cumulative (total) computation and communication costs for each participant while running Protocol 2 (both stage I and II) for the Enron email dataset, while Table 3 gives the costs for Protocol 3. Both tables report results for different polynomial sizes  $N$  and numbers of EDOs  $n$ . The size of a single ciphertext is  $2 \cdot N \cdot q$  bits. In Protocol 3, each participant transfers one ciphertext each, for a total (cumulative) communication cost of  $(n + 2)$  ciphertext transmissions ( $n$  EDOs, TACS and EDS), which are reported in the corresponding columns in Table 2 in MB. Table 4 gives the amortized (per query) costs of the improved 2PC version of Protocol 4 (Improvement 2 and 3 in Section 7) for each different  $N$ . In this sense, for the Enron dataset, the number of packed queries in a single ciphertext is  $q = \left\lfloor \frac{N}{(m+1)} \right\rfloor = \left\lfloor \frac{N}{(2047+1)} \right\rfloor = 4$  and 8 queries for  $N = 8192$  and  $N = 16,384$ , respectively, while for the SMS spam corpus dataset, it is 8 and 16 queries, respectively.

**Table 2.** Protocol 3 costs for different polynomial modulus  $N$  and EDO numbers  $n$ .

| $N$    | $n$ | Computation Cost (s) |       |     | Communication Cost (MB) |       |      |
|--------|-----|----------------------|-------|-----|-------------------------|-------|------|
|        |     | EDO                  | TACS  | EDS | EDO                     | TACS  | EDS  |
| 8192   | 10  | 197.7                | 394.4 | 2.0 | 190.2                   | 45.7  | 16.5 |
|        | 20  | 163.3                | 442.8 | 2.1 | 378.5                   | 59.3  | 22.5 |
|        | 30  | 148.5                | 484.4 | 2.1 | 566.4                   | 72.9  | 28.6 |
|        | 40  | 53.3                 | 176.6 | 2.5 | 754.3                   | 86.6  | 34.7 |
|        | 50  | 34.1                 | 173.6 | 1.9 | 482.1                   | 100.2 | 40.7 |
| 16,384 | 10  | 77.2                 | 141.6 | 1.5 | 379.3                   | 77.5  | 26.8 |
|        | 20  | 69.0                 | 195.5 | 1.1 | 756.6                   | 91.1  | 32.8 |
|        | 30  | 54.7                 | 240.7 | 2.3 | 1133                    | 104.8 | 38   |
|        | 40  | 39.2                 | 323.0 | 2.1 | 1510                    | 118.4 | 45.0 |
|        | 50  | 33.3                 | 386.5 | 2.1 | 1887                    | 132.0 | 51.0 |

**Table 3.** Protocol 3 costs for different  $N$  and numbers of EDOs.

| $n$                          | $N$   | Comp. (ms) $N = 8192$ |      |      | Com. | Comp. (ms) $N = 16,384$ |      |      | Com. |
|------------------------------|-------|-----------------------|------|------|------|-------------------------|------|------|------|
|                              |       | EDO                   | TACS | EDS  | (MB) | EDO                     | TACS | EDS  | (MB) |
| Enron email dataset [47]     |       |                       |      |      |      |                         |      |      |      |
| 10                           | 1690  | 149                   | 65   | 5.1  | 803  | 557                     | 53   | 20.5 |      |
| 20                           | 1382  | 212                   | 62   | 9.3  | 582  | 646                     | 73   | 37.6 |      |
| 30                           | 1044  | 198                   | 70   | 13.6 | 530  | 1581                    | 52   | 54.7 |      |
| 40                           | 941.0 | 203                   | 71   | 17.8 | 413  | 1966                    | 55   | 71.8 |      |
| 50                           | 283.3 | 235                   | 63   | 22.4 | 392  | 1752                    | 48   | 88.9 |      |
| SMS spam corpus dataset [54] |       |                       |      |      |      |                         |      |      |      |
| 10                           | 16.8  | 7.8                   | 4    | 5.1  | 36.5 | 34.9                    | 11.9 | 21.9 |      |
| 20                           | 16.7  | 9.5                   | 4.2  | 9.4  | 35.8 | 42.1                    | 1.7  | 40.2 |      |
| 30                           | 16.4  | 13.9                  | 3.7  | 13.7 | 36.1 | 53.4                    | 10.3 | 58.5 |      |
| 40                           | 16.3  | 17.4                  | 4.1  | 18.0 | 36.5 | 65.3                    | 10.8 | 76.8 |      |
| 50                           | 16.1  | 19.8                  | 3.9  | 22.3 | 37.8 | 71.0                    | 11.0 | 95.0 |      |

**Table 4.** Amortized (per query) costs of Protocol 4.

| Comput. Cost (ms)                    |      | Commun. Cost (KB) |     | Comput. Cost (ms)                      |      | Commun. Cost (KB) |     |
|--------------------------------------|------|-------------------|-----|----------------------------------------|------|-------------------|-----|
| EC                                   | EDS  | EC                | EDS | EC                                     | EDS  | EC                | EDS |
| Enron Email dataset [47], $N = 8192$ |      |                   |     | Enron Email dataset [47], $N = 16,384$ |      |                   |     |
| 15.03                                | 0.28 | 27.5              | 0   | 17.3                                   | 0.27 | 27.5              | 0   |
| SMS spam dataset [54], $N = 8192$    |      |                   |     | SMS spam dataset [54], $N = 16,384$    |      |                   |     |
| 6.5                                  | 0.2  | 13.75             | 0   | 6.95                                   | 0.72 | 13.75             | 0   |

In Table 5, we compare the best results of our secure feature-selection protocol (Protocol 2, both *secFS-S1* and *secFS-S2*) with the secure feature protocol in [40], which is the closest to ours from the related literature. In [40], they use a light version of the information gain—the Gini index—for secure feature selection. Also, in [40], they use the non-textual Speed Dating binary dataset of [58], which has  $NT = 8378$  records,  $|F| = 122$  initial features, from which they select  $m = 67$  features and finish the task in 60.57 min. Since [58] is a non-textual dataset and the features are known publicly, they do not engage in a protocol similar to our *GRPV* and *secFS-S1*, which makes their task easier.

In Tables 6 and 7, we report and compare the costs and characteristics of several related schemes (mainly related to binary textual datasets) dealing with PP training and classification, respectively. Since they report several costs and properties, we present the best of each one of each scheme. For the Enron dataset, in Table 6, in training costs of our scheme, we include both secure feature selection and training, compensating for the fact that the other schemes directly proceed with training without prior feature selection. For PP training and classification benchmark purposes with [29], for the SMS spam dataset [54,56] of our protocols, we report the results for  $N = 4096$ ,  $t = 27$  bits and  $q = 109$  bits and  $\lambda = 128$  bits.

Tables 5–7 show that our schemes outperform the others from several times to orders of magnitudes in terms of computation and communication, despite the others running their protocols on machines with much better hardware resources than ours. Our protocols are also better in classification accuracies.

**Table 5.** Secure feature selection comparisons among schemes.

| Scheme                    | $NT$   | $ F $   | $m =  SF $ | Comp.     | Comm.        |
|---------------------------|--------|---------|------------|-----------|--------------|
| Speed Dating dataset [58] |        |         |            |           |              |
| [40] *                    | 8378   | 122     | 67         | 60.57 min | Not reported |
| Enron email dataset [47]  |        |         |            |           |              |
| Ours                      | 33,703 | 157,458 | 2047       | 10.15 min | 310 MB       |

\* Four co-located F32s V2 Azure, each with 32 cores and 64 GB RAM.

**Table 6.** Training comparisons among similar PP schemes.

| Scheme                       | Comp. Cost | Comm. Cost | ML Algorithm  | Class. Acc. |
|------------------------------|------------|------------|---------------|-------------|
| Enron email dataset [47]     |            |            |               |             |
| [28] *                       | 11.1 days  | 120 GB     | Deep Learning | 86.3%       |
| [28] **                      | 5.04 days  | 120 GB     | Deep Learning | 86.3%       |
| Ours                         | 10.16 min  | 316 MB     | MNB           | 99.1%       |
| SMS spam corpus dataset [54] |            |            |               |             |
| [29]                         | 21.57 ms   | 763 KB     | NB            | 93.1%       |
| Ours                         | 18.20 ms   | 709 KB     | NB            | 93.1%       |
| MNIST dataset [59]           |            |            |               |             |
| [60]                         | 55.5 days  | NR         | Deep Learning | 96.3%       |

\* Utilizing 104 cores of Intel Xeon processors with 2.2 GHz and 482 GB of RAM. \*\* Improved version of the same scheme over the same hardware resources.

**Table 7.** Classification comparisons among similar PP schemes.

| Scheme                                                           | Comp. Cost      | Comm. Cost      | ML Algorithm    | Class. Acc.  |
|------------------------------------------------------------------|-----------------|-----------------|-----------------|--------------|
| Enron email dataset [47]                                         |                 |                 |                 |              |
| [14]                                                             | ≈8 s            | Not reported    | NB              | Not report.  |
| [15]                                                             | 350 ms          | ≈110 KB         | MNB, NB         | 98.8%        |
| [28] *                                                           | 3.79s (SEAL)    | 40.63 MB        | Deep Learning   | 86.3%        |
| [28] **                                                          | 0.17s (GPU)     | 40.63 MB        | Deep Learning   | 86.3%        |
| [32]                                                             | 78 min          | Not reported    | NB              | 99.1%        |
| <b>Ours</b>                                                      | <b>15.31 ms</b> | <b>27.5 KB</b>  | MNB             | <b>99.1%</b> |
| SMS spam corpus dataset [54]                                     |                 |                 |                 |              |
| [31]                                                             | 21 ms           | Not reported    | NB              | <b>95.6%</b> |
| [29]                                                             | 6.75 ms         | 109 KB          | NB              | 93.1%        |
| <b>Ours</b>                                                      | <b>3.38 ms</b>  | <b>11.37 KB</b> | NB              | 93.1%        |
| Hate speech against immigrants and women in Twitter dataset [30] |                 |                 |                 |              |
| [30]                                                             | 25.579 s        | Not reported    | Ensemble trees  | 74.4%        |
| [30]                                                             | 0.953 s         | Not reported    | Logis. Regress. | 72.4%        |

\* Twenty-six cores of 2.1 GHz Intel Xeon Platinum processor with 188 GB of RAM. \*\* 1 TESLA (5120 cores of 1.38 GHz) and 3 P100 (3584 cores of 1.19 GHz).

## 10. Security Analysis and Proofs

**Definition 7. Decision-RLWE:** for a security parameter  $\lambda$ , let  $f(x)$  be a cyclotomic irreducible polynomial of degree  $N$ , which depends on  $\lambda$ . Let  $Rq = \mathbb{Z}q[x]/f(x)$  be a polynomial ring with integer coefficients modulo  $q$ , s.t.  $q = q(\lambda) \geq 2$ . Let  $a, c$ , and  $s$  be uniform randomly chosen from  $Rq$ , i.e.,  $a \leftarrow Rq, c \leftarrow Rq$  and  $s \leftarrow Rq$ . Let  $e$  be a randomly chosen noise from a normal distribution  $\chi$ , i.e.,  $e \leftarrow \chi$ . Decision RLWE is the problem to distinguish between  $(a, a \cdot s + e)$  and  $(a, c)$  [51,52].

**Assumption 1. Hardness of Decision-RLWE:** Decision-RLWE is assumed to be a hard and resilient problem even for an adversary with a quantum computer [51,52].

**Semantic security of the RLWE schemes:** Due to its probabilistic encryption, RLWE-based schemes offer semantic security, i.e., for ciphertexts,  $m_{0-c}$  and  $m_{1-c}$  that encrypt plaintexts  $m_{0-p}$  and  $m_{1-p}$ , respectively; an adversary cannot distinguish which ciphertext belongs to which plaintext [61]. All the known SWHE schemes are based on RLWE.

**Definition 8. Secure Multi-Party Computation (SMPC) under the semi-honest model for deterministic functions:** Let us have  $p$  parties,  $P_1, \dots, P_p$ , with the corresponding private inputs  $x_1, \dots, x_p$  and let  $\bar{x} = (x_1, \dots, x_p)$ . With a certain security parameter,  $\lambda$ , let them execute protocol  $\Pi$  at the end of which each  $P_i$  gets the corresponding output  $O_{P_i}^\Pi(\lambda, \bar{x})$  for  $1 \leq i \leq p$ ; thus, the global output is  $O^\Pi = \left\{ \left( O_{P_i}^\Pi(\lambda, \bar{x}) \right)_{i=1}^p \right\}$ . Let the view of  $P_i$  be  $V_{P_i}^\Pi(\lambda, \bar{x}) = \left\{ \left( m_j^{P_i} \right)_{j=1}^t \right\}$ , where  $m_j^{P_i}$  are the messages that  $P_i$  receives while executing  $\Pi$ . We say that  $\Pi$  is a secure MPC protocol under the semi-honest model if there exists a simulator (function) s.t.  $S_{P_i}^\Pi(\lambda, x_i, O_{P_i}^\Pi(\lambda, \bar{x})) \cong_c V_{P_i}^\Pi(\lambda, \bar{x})$ , where  $\cong_c$  stands for computational indistinguishability against a probabilistic polynomial time adversary [12,61].

**Theorem 1. Modular Sequential Composition Theorem:** Let  $\Pi$  sequentially call protocols  $\Pi_1, \dots, \Pi_n$ . If  $\Pi_1, \dots, \Pi_n$  are SMPC protocols under the semi-honest model, then  $\Pi$  is also secure.

**Proof.** Given in [12,61].  $\square$

**Theorem 2. GRPV is secure under the semi-honest model.**

**Proof.** For  $1 \leq k \leq n$ ,  $V_{\text{EDO}_k}^{\text{GRPV}}(\lambda, \bar{x}) = \{\text{Enc}_{\text{pk}_k}(\text{rndVec}_v), \text{Enc}_{\text{pk}_k}(\text{rndVec}_{v_T})\}$  is the view and  $O_{\text{EDO}_k}^{\text{GRPV}}(\lambda, \bar{x}) = \{\text{hashBitPermVec}_v\}$  is the output of  $\text{EDO}_k$ .

Let  $\text{rndVec}_v$  be a random vector and let  $\text{rndVec}_{c_T} = \text{hashBitPermVec}_v - \text{rndVec}_v$ . For the simulator of each  $\text{eDo}$ , we have  $S_{\text{EDO}_k}^{\text{GRPV}}(\lambda, O_{\text{EDO}_k}^{\text{GRPV}}(\lambda, \bar{x})) = \{\text{Enc}_{\text{pk}_k}(\text{rndVec}_v), \text{Enc}_{\text{pk}_k}(\text{rndVec}_{c_T})\} \approx_c V_{\text{EDO}_k}^{\text{GRPV}}(\lambda, \bar{x})$ .

For the view of TACS, we have  $V_{\text{TEAS}}^{\text{GRPV}}(\lambda, \bar{x}) = \{(\text{rndVec}_{c_k})_{k=1}^n\}$ , and it has no output. For the simulator, we construct random SWHE ciphertexts; thus,  $S_{\text{TEAS}}^{\text{GRPV}}(\lambda) = (\text{rndVec}_{c_k})_{k=1}^n \approx_c V_{\text{TEAS}}^{\text{GRPV}}(\lambda, \bar{x})$  due to the semantic security of the RLWE SWHE schemes.

For EDS, the view is  $V_{\text{EDS}}^{\text{GRPV}}(\lambda, \bar{x}) = \{\text{rndVec}_c\}$ , and it has no output. For its simulator, we have  $S_{\text{EDS}}^{\text{GRPV}}(\lambda) = \{\text{rndVec}_c\} \approx_c V_{\text{EDS}}^{\text{GRPV}}(\lambda, \bar{x})$  since RLWE-based SWHE ciphertexts  $\text{rndVec}_c$  and  $\text{rndVec}_c$  are indistinguishable from each other due to their semantic security. None of the participants has a private input in GRPV.  $\square$

**Theorem 3. secFS-S1 is secure under the semi-honest model.**

**Proof.** For  $1 \leq k \leq n$ ,  $V_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, \bar{x}) = \{\text{Enc}_{\text{pk}_k}(W), \text{Enc}_{\text{pk}_k}(h2_v), \text{Enc}_{\text{pk}_k}(\text{rndWInDoc}_v)\}$  is the view (where  $\bar{x}$  the set of private input of all participants),  $x_{\text{EDO}_k}^{\text{secFS-S1}} = \{\text{PreProcess}(\text{DS}_k)\}$  is the private input, and  $O_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, \bar{x}) = \{\text{WGThanV}\}$  is the output of  $\text{EDO}_k$ . From  $x_{\text{EDO}_k}^{\text{secFS-S1}}$  and  $O_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, \bar{x})$ , we construct the set of the union of words  $\tilde{W} = \cup(W_k, \text{WGThanV}, \text{rndW})$ , where  $W_k$  is obtained as in lines 2–4 of *secFS-S1* and  $\text{rndW}$  are permuted hashes of random words.  $\tilde{h2}_v$  and  $\text{rndWInDoc}_v$  are randomly chosen such that in the resulting  $\text{rndWInDoc}_v - \tilde{h2}_v$ , the indexes that correspond to the words of  $\text{WGThanV}$  in  $\tilde{W}$  are positive, thus greater than  $\text{val}$ . For the simulator of  $\text{EDO}_k$ , we have  $S_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, x_{\text{EDO}_k}^{\text{secFS-S1}}, O_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, \bar{x})) = \{\text{Enc}_{\text{pk}_k}(\tilde{W}), \text{Enc}_{\text{pk}_k}(\tilde{h2}_v), \text{Enc}_{\text{pk}_k}(\text{rndWInDoc}_v)\}$ ; thus,  $S_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, x_{\text{EDO}_k}^{\text{secFS-S1}}, O_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, \bar{x})) \approx_c V_{\text{EDO}_k}^{\text{secFS-S1}}(\lambda, \bar{x})$ . For TACS,  $V_{\text{TEAS}}^{\text{secFS-S1}}(\lambda, \bar{x}) = \{(\text{Enc}_{\text{pk}_T}(W_k))_{k=1}^n, (\text{wordsInDoc}_{c_k})_{k=1}^n\}$  is the view,  $x_{\text{TEAS}}^{\text{secFS-S1}} = \emptyset$  is the input, and  $O_{\text{TEAS}}^{\text{secFS-S1}}(\lambda, \bar{x}) = \emptyset$  is the output it receives at the end of the protocol, where  $\emptyset$  is the empty set. Let us construct  $n$  sets of permuted word hashes  $(\tilde{W}_k)_{k=1}^n$  so that TACS cannot tell apart them and  $(W_k)_{k=1}^n$ , and  $n$  random ciphertexts  $(\text{wordsInDoc}_{c_k})_{k=1}^n$ , which cannot be distinguished from  $(\text{wordsInDoc}_{c_k})_{k=1}^n$  due to the semantic security of the RLWE schemes. For the TACS' simulator, then  $S_{\text{TEAS}}^{\text{secFS-S1}}(\lambda, x_{\text{TEAS}}^{\text{secFS-S1}}, O_{\text{TEAS}}^{\text{secFS-S1}}(\lambda, \bar{x})) = \{(\text{Enc}_{\text{pk}_T}(\tilde{W}_k))_{k=1}^n, (\text{wordsInDoc}_{c_k})_{k=1}^n\} \approx_c V_{\text{TEAS}}^{\text{secFS-S1}}(\lambda, \bar{x})$ .

EDS' view is  $V_{\text{EDS}}^{\text{secFS-S1}}(\lambda, \bar{x}) = \{\text{rndWInDoc}_c\}$ , and its input and output is the empty set. We construct a random ciphertext  $\text{rndWInDoc}_c$  which cannot be distinguished from  $\text{rndWInDoc}_c$  to the semantic security of the RLWE-based schemes. For the simulator, then we have  $S_{\text{EDS}}^{\text{secFS-S1}}(\lambda, x_{\text{EDS}}^{\text{secFS-S1}}, O_{\text{EDS}}^{\text{secFS-S1}}(\lambda, \bar{x})) = \{\text{rndWInDoc}_c\} \approx_c V_{\text{EDS}}^{\text{secFS-S1}}(\lambda, \bar{x})$ .  $\square$

**Corollary 1.** *secFS-S1 is secure under the semi-honest model when out of  $n$ ,  $n - 3$  EDOs fail or  $n - 2$  EDOs collude.*

**Proof.** Without a loss of generality, let the 3 left (not failed) EDOs be  $EDO_k$ , for  $1 \leq k \leq 3$ , and let us treat the  $n-2$  colluding EDOs as a single entity  $EDO_3$ ; thus, we again have 3 participating EDOs denoted as  $EDO_k$ , for  $1 \leq k \leq 3$ . We go through the similar lines as Theorem 2 to prove Corollary 1. If there are only 2 non-failed or non-colluding EDOs left, then, for the words in  $W$  (line 8 and 14), which do not appear in one EDO, not only this EDO will know that they exist in the other EDO, he will also know whether the counts of those words is greater or smaller than  $val$  based on the final  $WGThanV$  words' set, which constitutes a partial leakage that goes against the strict security and privacy requirements set in Section 1.  $\square$

**Theorem 4.** *secFS-S2 is secure under the semi-honest model.*

**Proof.** For  $1 \leq k \leq n$ ,  $V_{EDO_k}^{secFS-S2}(\lambda, \bar{x}) = \{Enc_{pk_k}(rndInforGains_v), Enc_{pk_k}(h1_v - h_v)\}$  is the view,  $x_{EDO_k}^{secFS-S2} = \{PreProcess(DS_k), WGThanV\}$  is the private input, and  $O_{EDO_k}^{secFS-S2}(\lambda, \bar{x}) = \{SF\}$  is the output of  $EDO_k$ . We construct random  $\widetilde{rndInforGains_v}$  and  $\widetilde{h1_v - h_v}$  s.t. after  $\widetilde{rndInforGains_v} - \widetilde{h1_v} - \widetilde{h_v}$ , the resulting vector, that in its corresponding indexes is supposed to store the relative IG of the words in  $WGThanV$ , which have its top  $m$  highest values for the indexes that correspond to the words of the selected features' set  $SF$ . For the corresponding simulator of each of the EDOs, we have  $S_{EDO_k}^{secFS-S2}(\lambda, x_{EDO_k}^{secFS-S2}, O_{EDO_k}^{secFS-S2}(\lambda, \bar{x})) = \{Enc_{pk_k}(\widetilde{rndInforGains_v}), Enc_{pk_k}(\widetilde{h1_v} - \widetilde{h_v})\}$ ; thus,  $S_{EDO_k}^{secFS-S2}(\lambda, x_{EDO_k}^{secFS-S2}, O_{EDO_k}^{secFS-S2}(\lambda, \bar{x})) \cong_c V_{EDO_k}^{secFS-S2}(\lambda, \bar{x})$ .

For TACS,  $V_{TEAS}^{secFS-S2}(\lambda, \bar{x}) = \{(\widetilde{hamWInDoc}_{c_k})_{k=1}^n, (\widetilde{spamWInDoc}_{c_k})_{k=1}^n, (\widetilde{hamM}_{c_k})_{k=1}^n, (\widetilde{spamM}_{c_k})_{k=1}^n, allTheRndLogs_c\}$  is the view,  $x_{TEAS}^{secFS-S2} = \emptyset$  is the input, and  $O_{TEAS}^{secFS-S2}(\lambda, \bar{x}) = \emptyset$  is the output. For the simulator, we construct the corresponding ciphertexts randomly, which cannot be differentiated from their counterparts due to the semantic security of the RLWE schemes; thus,  $S_{TEAS}^{secFS-S2}(\lambda, x_{TEAS}^{secFS-S2}, O_{TEAS}^{secFS-S2}(\lambda, \bar{x})) = \{(\widetilde{hamWInDoc}_{c_k})_{k=1}^n, (\widetilde{spamWInDoc}_{c_k})_{k=1}^n, (\widetilde{hamM}_{c_k})_{k=1}^n, (\widetilde{spamM}_{c_k})_{k=1}^n, allTheRndLogs_c\} \cong_c V_{TEAS}^{secFS-S2}(\lambda, \bar{x})$ .

EDS' view is  $V_{E2DS}^{secFS-S2}(\lambda, \bar{x}) = \{allRndNomDenom_c, rndInforGains_c\}$ , and its input and output are the empty set. For the simulator, we construct random ciphertexts, so  $S_{E2DS}^{secFS-S2}(\lambda, x_{E2DS}^{secFS-S2}, O_{E2DS}^{secFS-S2}(\lambda, \bar{x})) = \{allRndNomDenom_c, rndInforGains_c\} \cong_c V_{E2DS}^{secFS-S2}(\lambda, \bar{x})$ .  $\square$

**Corollary 2.** *secFS-S2 is secure under the semi-honest model when out of  $n$ ,  $n-3$  EDOs fail or  $n-2$  EDOs collude.*

**Theorem 5.** *secT is secure under the semi-honest model.*

**Proof.** For  $1 \leq k \leq n$ ,  $V_{EDO_k}^{secT}(\lambda, \bar{x}) = \emptyset$  is the view,  $x_{EDO_k}^{secT} = \{PreProcess(DS_k), SF\}$  is the private input, and  $O_{EDO_k}^{secT}(\lambda, \bar{x}) = \emptyset$  is the output of  $EDO_k$ . The simulator is  $S_{EDO_k}^{secT} = \emptyset$ .

For TACS,  $V_{TEAS}^{secT}(\lambda, \bar{x}) = \{(TV_{c_k})_{k=1}^n, rndTM_c\}$  is the view,  $x_{TEAS}^{secT} = \emptyset$  is the input, and  $O_{TEAS}^{secT}(\lambda, \bar{x}) = TM_c$  is the output. For the simulator, we construct random RLWE ciphertexts; thus,  $S_{TEAS}^{secT}(\lambda, x_{TEAS}^{secT}, O_{TEAS}^{secT}(\lambda, \bar{x})) = \{(TV_{c_k})_{k=1}^n, rndTM_c\} \cong_c V_{TEAS}^{secT}(\lambda, \bar{x})$ . For EDS,  $V_{E2DS}^{secT}(\lambda, \bar{x}) = \{rndTV_c\}$ ,  $S_{E2DS}^{secT} = \{rndTV_c\} \cong_c V_{E2DS}^{secT}(\lambda, \bar{x})$ .  $\square$

**Corollary 3.** *secT is secure under the semi-honest model when out of  $n$ ,  $n-3$  EDOs fail or  $n-2$  EDOs collude.*

**Theorem 6.** *Our end-to-end protocol (unprocessed datasets until the final trained model) is secure under the semi-honest model.*

**Proof.** The end-to-end protocol, denoted as  $E2E$ , sequentially calls  $GRPVS_{secFS-S1}$ ,  $secFS-S2$ , and  $secT$ , while their security was proven in Theorems 2–5, respectively. We invoke Theorem 1 to prove the security of  $E2E$ .  $\square$

**Corollary 4.**  *$E2E$  protocol is secure under the semi-honest model when out of  $n$ ,  $n - 3$  EDOs fail or  $n - 2$  EDOs collude.*

**Proof.** We use Corollaries 1–3 and invoke Theorem 1.  $\square$

**Theorem 7.**  *$secC$  is secure under the semi-honest model.*

**Proof.** EC's view is  $V_{EC}^{secC}(\lambda, \bar{x}) = \{rnd_{C_{TM}}(q)_{-v}, h2_{-v}\}$  its private input  $x_{EC}^{secC} = q_{-v}$  and output  $O_{EC}^{secC}(\lambda, \bar{x}) = C_{TM}(q)$ . For the EC's simulator, we construct random  $\widetilde{rnd_{C_{TM}}(q)_{-v}}, \widetilde{h2_{-v}}$  s.t. their subtraction will give the same output when EC executes lines 14–18; thus,  $S_{EC}^{secC}(\lambda, x_{EC}^{secC}, O_{EC}^{secC}(\lambda, \bar{x})) = \{\widetilde{rnd_{C_{TM}}(q)_{-v}}, \widetilde{h2_{-v}}\} \cong_c V_{EC}^{secC}(\lambda, \bar{x})$ . The views of TACS and EDS are  $V_{TEAS}^{secC}(\lambda, \bar{x}) = \{C_{TM}(q)_{-c}\}$  and is  $V_{EDS}^{secC}(\lambda, \bar{x}) = \{rnd_{C_{TM}}(q)_{-c}\}$ , respectively. We construct random ciphertexts  $\widetilde{C_{TM}(q)_{-c}}$  and  $\widetilde{rnd_{C_{TM}}(q)_{-c}}$  for their corresponding simulators.  $\square$

## 11. Discussions and Future Directions

We provide and prove the security of our end-to-end (secure feature selection proceeded by secure federated learning—training, i.e., from raw dataset to the final trained ML model), as well as secure classification protocol under the semi-honest model. In the process, we satisfy strict privacy and efficiency goals utilizing the properties offered by the proposed chain-of-blocks (COB) technology. Our secure protocols are robust and allow a high rate of fault tolerance or collusions among datasets, are independent of each other (can be used in combination with other schemes), are designed to have in mind distributed heterogeneous edge IoT environments and are resistant to adversaries with quantum computers. Furthermore, utilizing COB, the proposed protocols deal with network errors, as well as unauthorized and malicious edge dataset owners, which deviate from the protocol with the aim of disrupting it by eventually providing low classification accuracy. Experimental evaluations show that our protocols outperform state-of-the-art ones from several times to orders of magnitude in terms of computation and communication cost, while retaining the strict security and privacy requirements.

We plan to extend these security, privacy, and efficiency characteristics of our schemes to deal with secure multi-label multi-output ML algorithms in distributed environments, the first to do so in the literature, e.g., knowing that decision trees use the features' information gains to choose the nodes for each tree level, one such ML algorithm can be federated (distributed) tree learning, for which we can adjust our proposed protocol of securely evaluating the information gains in distributed environments ( $secFS-S2$ ). Furthermore, they will deal with horizontally, vertically, and hybrid partitioned datasets.

**Funding:** This research (including APC) was funded by İstanbul Sabahattin Zaim University grant number BAP-400-014.

**Data Availability Statement:** Codes, datasets, and experimental evaluations of the proposed protocols can be found in ref [55,56].

**Acknowledgments:** The author acknowledges the contributions of TÜBİTAK (The Scientific and Technological Research Institution of Turkey). The author would also like to thank Albert Levi and Erkey Savaş for their beneficial suggestions and remarks in improving the quality of this paper.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Yehuda, L.; Pinkas, B. Privacy preserving data mining. In *Annual International Cryptology Conference*; Springer: Berlin/Heidelberg, Germany, 2000.
2. Agrawal, R.; Srikant, R. *Privacy-Preserving Data Mining*; ACM: New York, NY, USA, 2000; Volume 29.
3. Verykios, V.S.; Bertino, E.; Fovino, I.N.; Provenza, L.P.; Saygin, Y.; Theodoridis, Y. State-of-the-art in privacy preserving data mining. *ACM Sigmod Rec.* **2004**, *33*, 50–57.
4. Aggarwal, C.C.; Yu, P.S. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-Preserving Data Mining*; Springer: Boston, MA, USA, 2008; pp. 11–52.
5. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.; Naehrig, M.; Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2016.
6. Bost, R.; Popa, R.A.; Tu, S.; Goldwasser, S. *Machine Learning Classification over Encrypted Data*; NDSS; Internet Society: Reston, VA, USA, 2015; Volume 4324.
7. Gao, C.-Z.; Cheng, Q.; He, P.; Susilo, W.; Li, J. Privacy-preserving Naive Bayes classifiers secure against the substitution-then-comparison attack. *Inf. Sci.* **2018**, *444*, 72–88.
8. Khedr, A.; Gulak, G.; Vaikuntanathan, V. SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. *IEEE Trans. Comput.* **2015**, *65*, 2848–2858.
9. Gupta, T.; Fingler, H.; Alvisi, L.; Walfish, M. Pretzel: Email encryption and provider-supplied functions are compatible. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, Los Angeles, CA, USA, 21–25 August 2017.
10. Kjamilji, A.; Idrizi, A.; Luma-Osmani, S.; Zenuni-Kjamilji, F. Secure Naïve Bayes classification without loss of accuracy with application to breast cancer prediction. In *Proceeding International Conference on Science and Engineering*; Sunan Kalijaga Press: Yogyakarta, Indonesia, 2020; Volume 3.
11. Wu, D.J.; Feng, T.; Naehrig, M.; Lauter, K. Privately evaluating decision trees and random forests. *Proc. Priv. Enhancing Technol.* **2016**, *2016*, 335–355.
12. Liu, X.; Lu, R.; Ma, J.; Chen, L.; Qin, B. Privacy-preserving patient-centric clinical decision support system on naive Bayesian classification. *IEEE J. Biomed. Health Inform.* **2015**, *2012*, 655–668.
13. Liu, X.; Deng, R.; Choo, K.-K.R.; Yang, Y. Privacy-preserving outsourced clinical decision support system in the cloud. *IEEE Transac. Serv. Comput.* **2017**, *14*, 222–234.
14. Al Badawi, A.; Hoang, L.; Mun, C.F.; Laine, K.; Aung, K.M.M. Privft: Private and fast text classification with homomorphic encryption. *IEEE Access* **2020**, *8*, 226544–226556.
15. Kjamilji, A.; Savas, E.; Levi, A. Efficient Secure Building Blocks with Application to Privacy Preserving Machine Learning Algorithms. *IEEE Access* **2021**, *9*, 8324–8353.
16. Merkle, C.R. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1987.
17. Narayanan, A.; Bonneau, J.; Felten, E.; Miller, A.; Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*; Princeton University Press: Princeton, NY, USA, 2016.
18. Li, S.; Xu, L.D.; Zhao, S. The internet of things: A survey. *Inf. Syst. Front.* **2015**, *17*, 243–259.
19. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge computing: Vision and challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646.
20. Androutsopoulos, I.; Paliouras, G.; Michelakis, E. *Learning to Filter Unsolicited Commercial E-Mail*; “DEMOKRITOS”, National Center for Scientific Research: Athens, Greece, 2004.
21. Ranaweera, P.; Jurcut, A.D.; Liyanage, M. Survey on Multi-Access Edge Computing Security and Privacy. *EEE Commun. Surv. Tutor.* **2021**, *23*, 1078–1124.
22. Zhao, W.; Jiang, C.; Gao, H.; Yang, S.; Luo, X. Blockchain-Enabled Cyber-Physical Systems: A Review. *IEEE Internet Things J.* **2020**, *8*, 4023–4034.

23. Das, K.; Bhaduri, K.; Kargupta, H. A local asynchronous distributed privacy preserving feature selection algorithm for large peer-to-peer networks. *Knowl. Inf. Syst.* **2010**, *24*, 341–367.
24. 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*; ACM: New York, NY, USA, 2011.
25. Jafer, Y.; Matwin, S.; Sokolova, M. Privacy-aware filter-based feature selection. In *2014 IEEE International Conference on Big Data (Big Data)*; IEEE: New York, NY, USA, 2014.
26. Sheikhalishahi, M.; Martinelli, F. Privacy-utility feature selection as a privacy mechanism in collaborative data classification. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*; IEEE: New York, NY, USA, 2017.
27. Rahimipour Anaraki, J.; Samet, S. Privacy-preserving feature selection: A survey and proposing a new set of protocols. *arXiv* **2020**, arXiv:2008.07664.
28. Rao, V.; Long, Y.; Eldardiry, H.; Rane, S.; Rossi, R.; Torres, F. Secure Two-Party Feature Selection. *arXiv* **2019**, arXiv:1901.00832.
29. Li, X.; Dowsley, R.; De Cock, M. Privacy-preserving feature selection with secure multiparty computation. *arXiv* **2021**, arXiv:2102.03517.
30. Reich, D.; Todoki, A.; Dowsley, R.; De Cock, M.; Nascimento, A.C.A. Privacy-preserving classification of personal text messages with secure multi-party computation: An application to hate-speech detection. *arXiv* **2019**, arXiv:1906.02325.
31. Resende, A.; Railsback, D.; Dowsley, R.; Nascimento, A.C.A.; Aranha, D.F. Fast privacy-preserving text classification based on secure multiparty computation. *arXiv* **2021**, arXiv:2101.07365.
32. Costantino, G.; La Marra, A.; Martinelli, F.; Saracino, A.; Sheikhalishahi, M. Privacy-preserving text mining as a service. In *2017 IEEE Symposium on Computers and Communic (ISCC)*; IEEE: New York, NY, USA, 2017.
33. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1999.
34. Tang, B.; Kay, S.; He, H. Toward optimal feature selection in naive Bayes for text categorization. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 2508–2521.
35. Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*; ACM: New York, NY, USA, 2009.
36. Martins, P.; Sousa, L.; Mariano, A. A survey on fully homomorphic encryption: An engineering perspective. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 83.
37. Smart, N.P.; Vercauteren, F. Fully homomorphic SIMD operations. *Des. Codes Cryptogr.* **2014**, *71*, 57–81.
38. Fan, J.; Frederik, V. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2012**, *2012*, 144.
39. Metsis, V.; Androutsopoulos, I.; Paliouras, G. *Spam Filtering with Naive Bayes-Which Naive Bayes?* CEAS: Mountain View, CA, USA, 2006; Volume 17.
40. Available online: <https://github.com/Microsoft/SEAL> (accessed on 20 May 2026).
41. Goldreich, O. Foundations of cryptography. In *Basic Applications*; Cambridge University Press: Cambridge, UK, 2009; Volume 2.
42. SMS Spam Collection v.1 Data Set. [Online]. Available online: <http://www.dt.fee.unicamp.br/~tiago/smsspamcollection> (accessed on 22 May 2021).
43. Available online: [https://github.com/artrimk/secNB\\_SMS\\_Bigrams](https://github.com/artrimk/secNB_SMS_Bigrams) (accessed on 20 May 2026).
44. Available online: [https://github.com/artrimk/secMNB\\_Email](https://github.com/artrimk/secMNB_Email) (accessed on 20 May 2026).
45. Available online: <https://github.com/OleanderSoftware/OleanderStemmingLibrary> (accessed on 20 May 2026).
46. Nandakumar, K.; Ratha, N.; Pankanti, S.; Halevi, S. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*; IEEE: New York, NY, USA, 2019.
47. Available online: <https://www.openml.org/d/40536> (accessed on 20 May 2026).
48. Kjamilji, A.; Levi, A.; Savas, E.; Güney, O.B. Secure Matrix Operations for Machine Learning Classifications over Encrypted Data in Post Quantum Industrial IoT. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)*; IEEE: New York, NY, USA, 2021.
49. Artrim, K. Privacy-Preserving Zero-Sum-Path Evaluation of Decision Tress in Postquantum Industrial IoT. *IEEE Trans. Ind. Inform.* **2024**, *20*, 10178–10191.
50. Kjamilji, A.; Güney, O.B. Highly efficient secure linear algebra for private machine learning classifications over malicious clients in the post-quantum world. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 101718.

51. Kjamilji, A. A Constant Time Secure and Private Evaluation of Decision Trees in Smart Cities Enabled by Mobile IoT. In 2023 *IEEE International Conference on Smart Mobility (SM)*; IEEE: New York, NY, USA, 2023.
52. Li, Q.; Ma, B.; Wang, X.; Wang, C.; Gao, S. Image steganography in color conversion. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *71*, 106–110.
53. Li, Q.; Wang, X.; Ma, B.; Wang, X.; Wang, C.; Gao, S.; Shi, Y. Concealed attack for robust watermarking based on generative model and perceptual loss. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *32*, 5695–5706.
54. Kjamilji, A. Secure, Private and Fast Memory Allocation of Processes in Post-Quantum Clouds. In 2026 *International Seminar on Intelligent Business and Edge-Computing Research (ISIBER)*; IEEE: New York, NY, USA, 2026.
55. Abia, P.; Huang, R. PriFidCE: An Efficient Privacy-Preserving Face Identification Protocol from a More Compact Coefficient Embedding. *IEEE Internet Things J.* **2025**, *13*, 8601–8614.
56. Kjamilji, A. Privacy Preserving Tax Evasion Detection for Post-Quantum Centralized Banking Systems. In 2025 *1st International Conference on Emerging Trends in Information Systems and Informatics (ICETISI)*; IEEE: New York, NY, USA, 2025.
57. Kjamilji, A. Covert, Secure and Private Communications in Software Defined Networking. In 2023 *International Conference on Computational Science and Computational Intelligence (CSCI)*; IEEE: New York, NY, USA, 2023.
58. Li, D.; He, K.; Liu, J.-N.; Yang, A.; Liu, X.; Weng, J. PDTE: Outsourced Private Decision Tree Evaluation for Large Trees with Sublinear Communication. *IEEE Trans. Serv. Comput.* **2026**. <https://doi.org/10.1109/TSC.2026.3689639>.
59. Li, D.; Chattopadhyay, A.; Li, Q.; Lü, Q.; Wu, J.; Xiang, T.; Liao, X. CryptDNN: A Fast Privacy-Preserving Deep Neural Network Inference Architecture Based on Cloud-Edge-Client Collaboration. *IEEE Trans. Netw. Sci. Eng.* **2025**, *13*, 2726–2740.
60. He, G.; Yuan, Z. Verifiable Privacy-Preserving Neural Network Inference via Multi-Key Homomorphic Encryption. *IEEE Internet Things J.* **2026**, *13*, 22072–22081.
61. Zeng, B.; Wu, T.; Peng, S.; Pang, Z.; Yu, R.; Wang, L. Efficient privacy-preserving convolutional neural network inference with skip-perm. *Adv. Eng. Inform.* **2026**, *69*, 104081.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.