applied sciences

MDPI

Article

A Privacy-Preserving Scheme for a Traffic Accident Risk Level Prediction System

Pablo Marcillo *, Gabriela Suntaxi and Myriam Hernández-Álvarez

Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Ladrón de Guevara E11-25 y Andalucía, Edificio de Sistemas, Quito 170525, Ecuador; gabriela.suntaxi@epn.edu.ec (G.S.); myriam.hernandez@epn.edu.ec (M.H.-Á.)
* Correspondence: pablo.marcillo@epn.edu.ec

Abstract: Due to the expansion of Artificial Intelligence (AI), especially Machine Learning (ML), it is more common to face confidentiality regulations about using sensitive data in learning models generally hosted in cloud environments. Confidentiality regulations such as HIPAA and GDPR seek to guarantee the confidentiality and privacy of personal information. Input and output data of a learning model may include sensitive data that must be protected. Adversaries could intercept and exploit this data to infer more sensitive data or even to determine the structure of the prediction model. To guarantee data privacy, one option could be encrypting data and making inferences over encrypted data. This strategy would be challenging for learning models that now must receive encrypted data, make inferences over encrypted data, and deliver encrypted data. To address this issue, this paper presents a privacy-preserving machine learning approach using Fully Homomorphic Encryption (FHE) for a model that predicts risk levels of suffering a traffic accident. Despite the limitations of experimenting with FHE on machine learning models using a low-performance computer, limitations that are undoubtedly overcome by using high-performance computational infrastructure, we built some encrypted models. Among the encrypted models based on Decision Trees, Random Forests, XGBoost, and Fully Connected Neural Networks (FCNN), the model based on FCNN reached the highest accuracy (80.1%) for the lowest inference time (8.476 s).

Keywords: security; fully homomorphic encryption; privacy-preserving machine learning; traffic accident prevention; fully connected neural network

Citation: Marcillo, P.; Suntaxi, G.; Hernández-Álvarez, M. A Privacy-Preserving Scheme for a Traffic Accident Risk Level Prediction System. *Appl. Sci.* **2024**, *14*, 9876. https://doi.org/10.3390/app14219876

Academic Editor: Young-Gab Kim

Received: 6 August 2024
Revised: 22 September 2024
Accepted: 26 September 2024
Published: 29 October 2024

1. Introduction

The Internet of Things (IoT) has evolved in such a way that almost every gadget used in everyday life has embedded sensors and communication interfaces to exchange information. Currently, IoT gadgets can generate enormous quantities of data in real-time. The emergence of big data and AI has made it possible to use these data to detect trends, recognize patterns, and predict behaviors related to topics of interest. Until a few years ago, data analysis and its applications covered the challenges and expectations in technology; however, learning from data has gathered strength and has become a dominant tool in AI.

At present, it is easy to find machine-learning models for areas such as genetics, healthcare, medical informatics, finances, entertainment, and transportation. In the latter, there are prediction models for vehicular traffic flow, rail track degradation, vehicle traffic accidents, and others [1]. The issue of preventing traffic accidents has generated great interest in the community because of the significant impact that research in this area could have. Thus, having a learning model to predict risk levels of suffering an accident would be a great contribution to the general population.

Traffic accidents are not casual events but the sum of factors given in a certain space and time [2]. Some authors have mentioned serious factors in traffic accidents. For instance, the authors in [3,4] mentioned unfavorable traffic characteristics, adverse weather conditions, driver distraction, and aggressive driving after unusual congestion, and the authors in [5] confirmed that road geometric characteristics and traffic flow directly impact the number of traffic accidents in road sections. Other authors [6] have even gone further in determining the severity of a traffic accident using driver and vehicle information, weather and light conditions, and road conditions. Therefore, as the learning models use more data from different sources, they will be more precise in inferring the risk of suffering a traffic accident. For instance, our Risk Level Prediction System (Section 4.1) uses data from five heterogeneous sources: driver and vehicle information, weather conditions, traffic accidents, and road geometric characteristics.

Despite the benefits of learning models, some solutions have been discarded because of restrictions on using certain information. For example, solutions in healthcare have been affected by confidentiality regulations regarding patients' medical records [7]. The data often contain sensitive information that must be protected from unauthorized access. Using sensitive information in machine learning models without a data privacy scheme could cause information leakage and associated consequences. Adversaries can intercept sensitive information such as the vehicle location, timestamp, or driver's ID and determine routes and schedules used by them. Then, we aim to propose a privacy-preserving scheme that guarantees data privacy even in the presence of adversaries for predicting traffic accident risk levels.

As is well known, the default mechanism to guarantee data privacy is encryption. Encrypting sensitive information using traditional encryption methods can be effective; however, that information must be deciphered to a certain point to train the models and make inferences, which could result in a serious data privacy issue. In that sense, Homomorphic Encryption (HE) provides a solution to ensure the privacy of both information and models. Based on this requirement, the Privacy-Preserving Machine Learning (PPML) research area has gained interest in the machine learning community.

Our proposal consists of a privacy-preserving scheme for a system that includes a learning model that predicts the risk levels of suffering a traffic accident, which is built under an adversary model and uses Homomorphic Encryption. The adversary model identifies and describes all adversaries who could take advantage of capturing sensitive information. Considering those adversaries, our proposal involves processing encrypted data. In other words, our proposal requires performing operations on encrypted data without decrypting it before using it in the learning model. Additionally, data must be encrypted end-to-end and kept encrypted at any time. Thus, we chose Homomorphic Encryption, specifically Fully Homomorphic (FHE), because it offers unlimited operations of additions and multiplications over encrypted data, unlike other encryption methods.

Lastly, our contribution is a privacy-preserving scheme based on fully Homomorphic Encryption for a system that includes a learning model that predicts risk levels of suffering a traffic accident. It includes a circuit for key generation and inference, where a secret key is generated for the encryption and decryption of data and results, an encrypted model for encrypted inference, and an evaluation key for performing operations on encrypted data. This circuit guarantees that the data and the model are always protected, improving their privacy and security. In summary, the main idea is a PPML model that can infer directly from encrypted data and produce encrypted results. Our proposal prevents adversaries described in the adversary model from replicating models, identifying statistical characteristics of data, or inferring inputs or outputs of the learning model [8,9].

The rest of this paper is organized as follows. Section 2 presents the state of the art of this topic. Section 3 presents definitions, concepts, and ideas on Homomorphic Encryption and security notions. Section 4 describes the initial system, problem definition, adversary model, our solution, and secrecy proofs. Section 5 presents the design of the experiments that validate our proposed scheme. Section 6 presents the results of the evaluation of our privacy-preserving security schema. Finally, Section 7 discusses the most relevant thoughts about this topic, and Section 8 presents the conclusions of this work.

2. Related Work

Until a few years ago, the main concerns of authors when building learning models were the availability of data and, later, the management of a large amount of data. Those limitations were solved with the spreading of IoT and the emergence of Big Data; however, the current main concern is the restriction on using sensitive information in the learning models. Thus, much of the recent research is focused on looking for strategies or mechanisms to protect sensitive information used by the learning models. One of them is the implementation of Homomorphic Encryption in machine learning.

The proposals related to this topic are spread across different research areas and use different types of Homomorphic Encryption. Table 1 summarizes the most important of the proposals. In Genetics and Genomics, the authors in [10,11] used HE as the type of encryption in their proposals. Thus, Hong et al. [10] proposed a secure multi-label tumor classification model based on a Homomorphic Encryption scheme. This model uses the softmax activation function and efficient data encryption. Kim et al. [11] proposed a solution to train a HE-based logistic regression model to identify cardiopathies. The homomorphic scheme works with real numbers and uses an encoding method for reducing encrypted storage.

In the field of medical informatics, the authors in [12,13] used HE, and those in [7,14] used FHE as the type of encryption. Thus, Kim et al. [12] proposed a Homomorphic Encryption scheme for learning models. This scheme allows for creating an encrypted logistic regression-based model optimized for real number computation and large-scale datasets. Popescu et al. [13] proposed a privacy-preserving classification model of EEG signals for seizure detection and alcohol predisposition using HE. Their model uses a HE-based encoding method that also works with real numbers. Vizitiu et al. [7] proposed a variant of the Matrix Operation for Randomization and Encryption (MORE) scheme for deep neural networks. Its applicability to deep learning models has focused on resolving digit recognition, blood flow circulation, and coronary angiography classification.

In the field of computer science, the authors in [15–17] used FHE, those in [9,18,19] used HE, and those in [20] used Partially Homomorphic Encryption (PHE) as the type of encryption. Thus, Wu et al. [15] proposed a secure and efficient clustering scheme based on K-Means, FHE, and a ciphertext packing technique that permits parallel computation. This scheme preserves privacy in database security, clustering results, and data access patterns to prevent inference attacks. The evaluation of the scheme shows a low computational cost, making it more efficient compared with similar models. Sun et al. [16] proposed an FHE-based secure classification model based on Decision Trees. This model supports encoding for real numbers and uses re-linearization to reduce multiplicative ciphertext and modulus switching techniques for decreasing multiplicative ciphertext modules and decryption noise. Lee et al. [17] proposed a privacy-preserving scheme based on FHE for classifying images using neural networks. This scheme incorporates Rectified Linear Unit (ReLU) and Softmax as activation functions but with approximation methods to evaluate them and bootstrapping of the RNS-CKKS as the resampling technique. They applied the scheme to a RestNet-20 model based on FHE. Li et al. [9] proposed a Homomorphic Encryption scheme based on non-commutative rings for privacy-preserving machine learning. This scheme uses the conjugacy search problem to guarantee one-way security. They also proposed Homomorphic Encryption over a ring of square matrices, which supports the encryption of real numbers. Regarding the performance, the authors mention that their scheme is efficient at the moment of encryption and decryption of information and performing homomorphic operations. Sarkar et al. [20] proposed a genotype imputation technique applied to PHE-based linear models. They also introduced optimization for data sequence prediction for genotype imputation. After evaluating linear models and their privacy-preserving equivalents, the results show similar performance for both, even using large-scale datasets. Cheon et al. [18] proposed a method to optimize the iterations of the gradient descent algorithm. They implemented the method for logistic regression models based on HE for the learning phase. This method considerably reduces the number of iterations of the optimization algorithms, which results in a reduction of time of the learning process of the encrypted model and a reduction of the storage of encrypted data.

Finally, the following proposals provide models applied indistinctly to different areas (financial, medical informatics, and computer science). All three proposals [8,21,22] use FHE as the type of encyption. Thus, Bajard et al. [8] proposed an FHE-based Support Vector Machine (SVM) model that includes efficient techniques for polynomial and sign evaluation. The polynomial evaluation minimizes the multiplicative depth of the circuit, and the sign evaluation improves the sign approximation by proposing a way to randomize the iterations. These techniques have improved the value-wise and bit-wise arithmetic. In addition, Park et al. [21] proposed an algorithm for an SVM model based on gradient descent with support to FHE. This algorithm was designed for the training phase and to protect both the model and the training data. The encryption scheme for this approach supports computations on real numbers.

In addition to work in encryption, several secure schemes have been developed in recent years to predict traffic accidents using non-encryption-based methods. The authors in [23] proposed an approach that involves access control and authentication protocols, ensuring that only authorized entities can access sensitive traffic and vehicle data. Some studies propose using blockchain to establish a decentralized, tamper-proof system for storing and sharing traffic data, which can be leveraged for accident prediction while maintaining integrity and transparency [24]. Liu et al. [25] introduced a federated learning-based framework to ensure privacy in traffic flow prediction by aggregating model parameters rather than sharing raw data.

Considering our requirements and the purpose of this work, the analyzed proposals present limitations. Thus, [9,14–16,18,20] did not include experiments in which encrypted models using Homomorphic Encryption were evaluated and, therefore, did not provide accuracy values. Other proposals [8,10,11,19,22] did not present accuracy values for the unencrypted models to compare their results. In general, none of the proposals were focused on areas of study such as transportation or similar, specifically road traffic safety. In Refs. [9–13,18], the authors did not clearly define the Homomorphic Encryption scheme used, or they mentioned it in general terms. Next, some proposals [7,13,17,21] did not include an adversary model because they aim to apply Homomorphic Encryption to learning models rather than provide a privacy-preserving scheme based on HE. Finally, while the non-encryption-based approaches provide useful mechanisms for ensuring privacy and data security in traffic accident prediction, they face several limitations compared to FHE For instance, methods like access control [23], blockchain [24], and federated learning [25] still expose certain data elements, such as model parameters or metadata, to potential inference attacks. In contrast, FHE offers stronger privacy guarantees by allowing computations to be performed directly on encrypted data, ensuring that sensitive information remains fully encrypted throughout the process without any data exposure. Additionally, FHE mitigates the communication overhead and accuracy degradation challenges seen in federated learning models, providing both superior privacy and efficiency in accident prediction.

Table 1. Proposals of privacy-preserving models using homomorphic encryption

Authors	**Algorithm**	**Encr.**	**Library**	**Purpose**	**Accuracy**		**Area of Study**
					Decr.	**Encr.**	
Hong et al. [10]	Neural Network	HE	-	Secure tumor classification	-	85.15	Genetics, Genomics, and Proteomics
Kim et al. [12]	Logistic Regression	HE	HELR [26]	Classification of			Medical Informatics
				•Myocardial infarction	88.43	86.03	
				•Low birth weight	68.25	69.30	
				•Physical health status and nutrition	79.26	79.23	
				•Prostate cancer	68.86	68.85	
				•Drug abuse treatment	74.43	74.43	

Table 1. *Cont.*

Authors	Algorithm	Encr.	Library	Purpose	Accuracy Decr.	Accuracy Encr.	Area of Study
Wu et al. [15]	K-Means	FHE	Seal [27]	Privacy-preserving in database security, clustering, and data access patterns	-	-	Comput. Sci.
Sun et al. [16]	Decision Tree	FHE	HElib [28]	Private machine learning	-	-	Comput. Sci.
Lee et al. [17]	Neural Network	FHE	Seal	Privacy-preserving machine learning	91.89	92.43	Comput. Sci.
Popescu et al. [13]	Polynomial Regression	HE	-	Privacy-preserving classification model for			Medical Informatics
				•Seizure detection	92.26	92.26	
				•Alcohol predisposition	79.83	79.83	
Li et al. [9]	Logistic Regression, Naive Bayes, SVM, Decision Tree, and Random Forest	HE	-	Homomorphic encryption scheme based on non-commutative rings	-	-	Comput. Sci.
Kim et al. [11]	Logistic Regression	HE	HEAAN [29]	Training of a logistic regression model based on Approximate HE for identification of cardiopathies	-	61.72	Genomics
Bajard et al. [8]	SVM	FHE	LIBSVM	Techniques for improving efficiency in SVM models based on FHE for			Financial, Medical Informatics, Comput. Sci.
				•Credit card approval	-	86.32	
				•Sonar signals	-	89.58	
				•Diabetes diagnosis	-	76.79	
				•Heart Disease	-	81.43	
Park et al. [21]	SVM	FHE	HEAAN	FHE-based algorithm for a SVM model for			Financial, Medical Informatics, Comput. Sci.
				•Liver disorders	73.00	73.00	
				•Credit risks	69.00	69.00	
				•Heart disease	85.00	85.00	
				•Diabetes diagnosis	73.00	73.00	
				•Sonar signals	88.00	88.00	
Sarkar et al. [20]	Logistic Regression and Neural Network	PHE	-	Privacy-preserving genotype imputation technique	-	-	Comput. Sci.
Cheon et al. [18]	Logistic Regression	HE	HEML [29]	Ensemble method based on HE for logistic regression	-	-	Comput. Sci.
Vizitiu et al. [7]	Neural Network	FHE	-	Privacy-preserving deep learning	98.20	98.20	Medical Informatics
Han et al. [22]	Logistic Regression	FHE	-	Logistic regression on fully homomorphically encrypted data applied to			Financial and Comput. Sci.
				•Credit rating	-	80.00	
				•Image processing	-	96.40	
Marcano et al. [14]	Convolutional Neural Network	FHE	-	Privacy-preserving model for image recognition	-	-	Comput. Sci
Han et al. [19]	Naive Bayes	FHE	HEAAN	Privacy-preserving classifier for breast cancer	-	97.80	Medical Informatics

3. Preliminaries

In this section, we present information on homomorphic encryption and security notions.

3.1. Homomorphic Encryption

HE permits computation directly on encrypted data without deciphering it [30]. Generally, an HE scheme has four functions: *KeyGen*, *Enc*, *Dec*, and *Eval* [31]. First, the function *KeyGen* generates the public and secret keys. The equation is presented as follows:

$$(pk, sk) \leftarrow KeyGen() \tag{1}$$

where *pk* is the public key and *sk* is the secret key.

The function *Enc* performs an encryption operation. The following equation represents the transformation from plaintext into ciphertext:

$$c \leftarrow Enc(pk, p) \tag{2}$$

where c is a ciphertext, pk is the public key, and p is a plaintext.

The function *Dec* performs a decryption operation. The following equation represents the transformation from ciphertext into plaintext:

$$p \leftarrow Dec(sk, c) \tag{3}$$

where p is a plaintext, sk is the secret key, and c is a ciphertext.

HE schemes also have the function *Eval*. This function evaluates a circuit (e.g., a machine learning model). The equation is presented as follows:

$$Eval(pk, C, c0, \ldots, cn) \tag{4}$$

where pk is the public key, C is a circuit, and $c0, \ldots, cn = Enc(pk, p0, \ldots, pn)$.

As a result, the following equation establishes that the result of circuit C on plaintext is equal to performing the *Enc*, *Eval*, and *Dec* functions:

$$Dec(sk, Eval(pk, C, Enc(pk, p0, \ldots, pn))) = C(p0, \ldots, pn) \tag{5}$$

Depending on the type and number (depth) of operations, HE schemes can be classified as Partially Homomorphic Encryption, Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption. The following is a short description of the schemes mentioned above.

3.1.1. Partially Homomorphic Encryption

PHE allows performing an unlimited number of a single type of allowed operations on encrypted data. The algorithms used for applications in which PHE schemes are applied support only additions or multiplications, not both. Among the major PHE schemes are RSA, GM, El-Gamal, Benaloh, NS, OU, Pailier, DJ, KTX, and Galbraith [32].

3.1.2. Somewhat Homomorphic Encryption

SHE allows performing a limited number of available evaluation operations. The weakness of this scheme is that the maximum number of operations is limited because the size of the ciphertext increases with each operation. Concerning SHE schemes, the major ones are Yao, SYY, BGN, and IP [32].

3.1.3. Fully Homomorphic Encryption

FHE permits performing an unlimited number of operations from the available ones. Different research areas have received FHE because of its applicability in real-life applications. Also, because of its mathematical concepts, implementing FHE schemes is complex. The major schemes are Ideal Lattice-based, Over Integers, RLWE-based, and NTRU-like [32].

Section 4.3 will mention the type of Homomorphic Encryption used in our privacy-preserving scheme and the justification for selecting it.

3.2. Security Notions

A security notion guarantees that any cryptographic construction to be implemented is secure enough given a certain environment. A security notion combines a security goal and an attack model. The following are the major security goals.

3.2.1. Indistinguishability

Indistinguishability is a security notion that establishes whether a Public-Key Encryption (PKE) is secure. If a cryptosystem has this notion, the adversary cannot distinguish cyphertexts from plaintexts. In cryptography, there are some security notions in terms of indistinguishability [33].

- Indistinguishability under Chosen-Plaintext Attack (IND-CPA): This is the most basic security notion. In this case, a passive (computationally limited) adversary, given a pk, cannot distinguish the cyphertexts of a pair of messages.
- Indistinguishability under Chosen-Ciphertext Attack (IND-CCA): This is the standard security notion for PKE. In this case, an active adversary, given a pk and even a decryption oracle, cannot distinguish the cyphertexts of a pair of messages.

3.2.2. Non-Malleability

Non-Malleability is a security goal in which the adversary cannot create a new cyphertext ($c2$) from an original cyphertext ($c1$) that comes up with a new plaintext ($p2$) that is meaningfully related to the original plaintext ($p1$). Similar to Indistinguishability, there are some security notions in terms of Non-Malleability based on the CPA and CCA attacks: NM-CPA and NM-CCA.

4. Materials and Methods

In this section, we present a risk level prediction system for traffic accident prevention, its security risks, and a security approach for it.

4.1. Risk Level Prediction System for Traffic Accident Prevention

Our system is based on in-vehicle and infrastructure domains. The in-vehicle domain consists of Onboard Units (OBUs) and Application Units (AUs) installed in the vehicles, and the infrastructure domain consists of vehicles equipped with OBUs and AUs that can communicate with external networks through Roadside Units (RSUs). This system uses Amazon Web Service (AWS) services for storage, processing, model training, and deploying information. This system consists of three components or modules called agents. Those agents are built on vehicular scanners, Global Positioning System (GPS) receptors, heart rate and oxygen saturation monitors, cloud services, and mobile apps. Figure 1 presents the architecture of the Risk Level Prediction System.

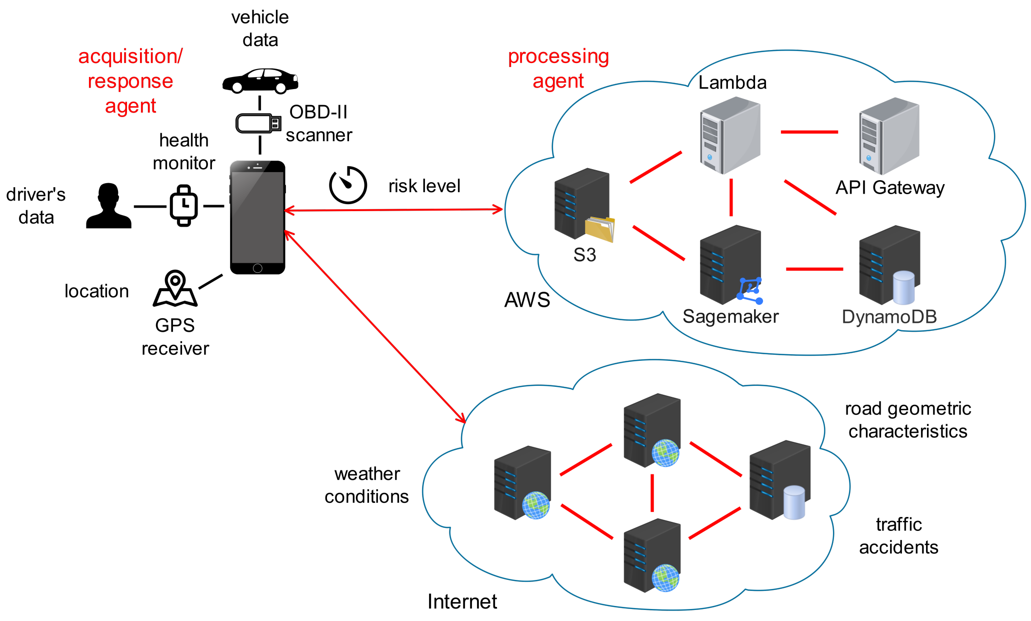

Figure 1. Architecture of the Risk Level Prediction System.

The Risk Level Prediction System consists of three main agents: acquisition, processing, and response.

First, the acquisition agent is a mobile app that recollects data from an On-board Diagnostic (OBD) scanner, GPS receptor, and a smartwatch to track the driver's heart rate. It also retrieves information from a weather service and a traffic accident database to collect data on weather conditions and the number of accidents in a specific location. The acquisition agent uses the current releases of AccuWeather, S3, and DynamoDB services.

Second, the processing agent is a cloud-based tool that processes driving data (driver and vehicle information, weather conditions, traffic accidents, and road characteristics) and returns the risk level through machine learning models. The processing agent uses the current releases of S3, Sagemaker, and DynamoDB services.

Finally, the response agent is a mobile app that mainly retrieves the traffic accident risk level from the processing agent and presents it to the drivers using notifications and alerts. The response agent relies on the current release of DynamoDB service.

4.2. The Privacy-Preserving Scheme

4.2.1. Problem Definition

Considering the system architecture, vehicles, roadside units, and cloud services exchange messages with each other. Malicious adversaries can intercept those messages, which include sensitive information such as date, time, location data, vehicle ID, driver's ID, and more information that can be inferred. For instance, adversaries can determine the routes and schedules used by their victims using vehicle location and the message timestamp. Adversaries could identify the vehicle and establish a driver's profile. This information could even be sold to criminals who could use it to kidnap the driver and steal the vehicle in the worst-case scenario.

4.2.2. Adversary Model

According to the system architecture (Figure 2), the adversaries can be located within or outside of the Vehicular Ad-Hoc Network (VANET), in any other network (e.g., a mobile network), or the cloud service network. In other words, they can be part or not of a network and can be an authorized or unauthorized user. Depending on the location of the adversaries, they can perform different attacks. For instance, when adversaries are in the middle of the communication between the vehicles and roadside units, they can perform a Man-in-the-Middle (MitM) attack. Similarly, they can perform a replay attack by eavesdropping the communication channel to intercept messages, which they will then fraudulently resend. Additionally, they can perform a tampering attack by maliciously altering the messages exchanged between vehicles and roadside units.

Our adversary model considers adversaries Type I, II, III, and IV with their assumptions and capabilities. These adversaries can act like honest but curious or malicious users. Table 2 presents the adversary model. A description of those types is presented as follows.

- Type I is an internal adversary who is an authorized user of a VANET. They can act like an honest and curious adversary and listen to the communication channel to intercept messages.
- Type II is an external adversary who is not an authorized user of a VANET and is located between the VANET and a roadside unit. They can perform Tampering, Impersonation, MitM, Denial of Service (DoS), Replay, and other attacks.
- Type III is an external adversary who uses any other network, for instance, the Internet. They can perform DoS, Malware Injection, and other attacks.
- Type IV is an internal adversary who is an authorized user of the cloud service provider network. Similarly to Type 1, they can act like an honest and curious adversary.

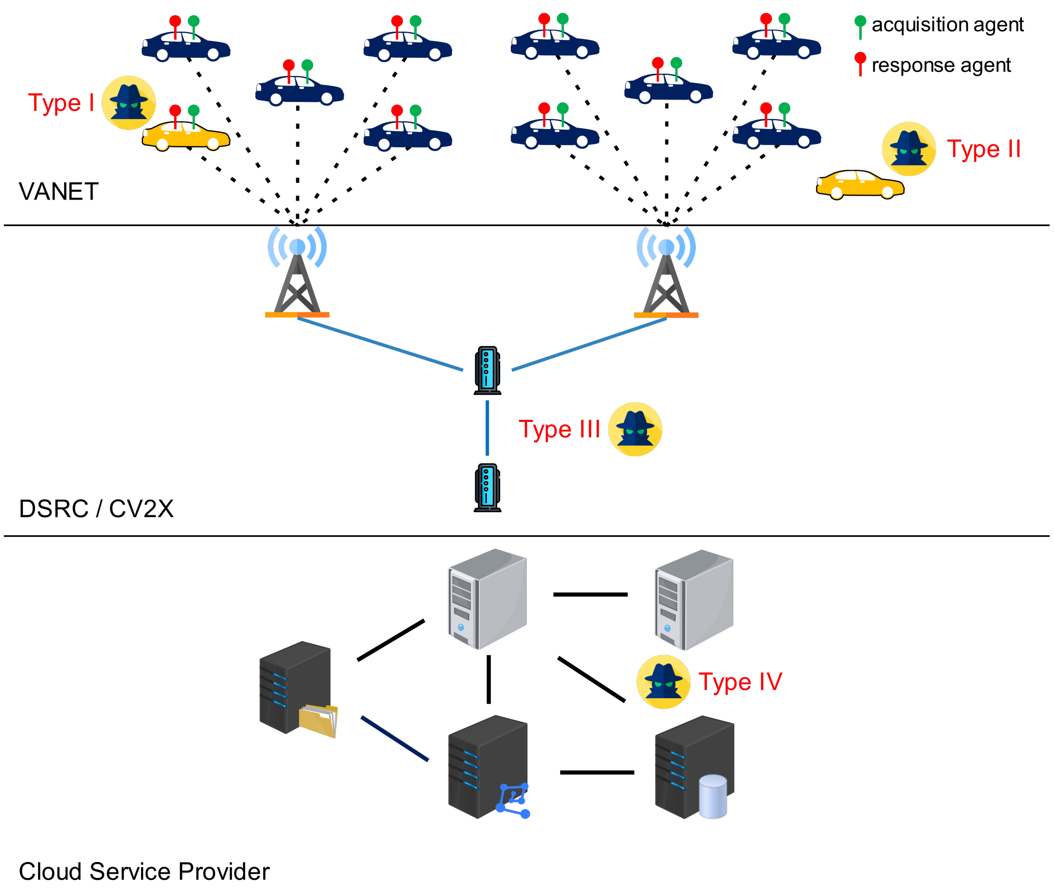

Figure 2. Location of adversaries into the system architecture.

Table 2. Adversary model.

Class	Scope	Target	Assumptions	Capabilities
Type I	Internal	VANET	Member of the vehicular network	They can only listen to channel communication and intercept messages
Type II	External	VANET	Not a member of the vehicular network	They can perform Tampering, Impersonation, MitM, DoS, Replay, and other attacks.
Type III	External	DSRC/CV2X	Member of the any other network	They can perform DoS, Hijacking, Malware Injection, Side-Channel, Man-in-the-Cloud, and other attacks.
Type IV	Internal	Cloud Service Provider	Member of the cloud service provider network	They can only listen to channel communication and intercept messages.

4.3. Our Security Approach

Considering that, first, approximately one of every two proposals use any cryptography to improve their security [34], and second, the messages circulating in the telecommunications infrastructure include sensitive information, and also taking into account that adversaries with specific capabilities can compromise security requirements, we consider the cryptographic mechanisms as a solution to guarantee the confidentiality and integrity of the information.

We initially considered using public key cryptography. While it could work, there is a risk of authorized adversaries within the cloud service provider network intercepting sensitive information once it is unencrypted to be processed. Then, we analyzed the implementation of a cryptography computing technique, specifically HE, as a solution. Despite the high computational cost of using HE, we chose this technique because we need to perform operations over encrypted data. Also, using this type of cryptography would prevent internal or external adversaries from seeing the content of the messages, which is possible because the processing would be done over encrypted data. Thus, our privacy-preserving scheme will be based on Homomorphic Encryption, specifically Fully Homomorphic Encryption. We chose FHE rather than Partially or Somewhat Homomorphic Encryption because it can perform unlimited evaluation operations from many available operations.

Referring to the Risk Level Prediction System for Traffic Accident Prevention, our scheme will permit encrypting the outgoing data provided by the acquisition agent, processing that encrypted data through the processing agent with its encrypted learning model, and sending back the encrypted result to the response agent. This circuit is performed without decrypting the data at any time. Figure 3 presents our privacy-preserving scheme for the learning model.

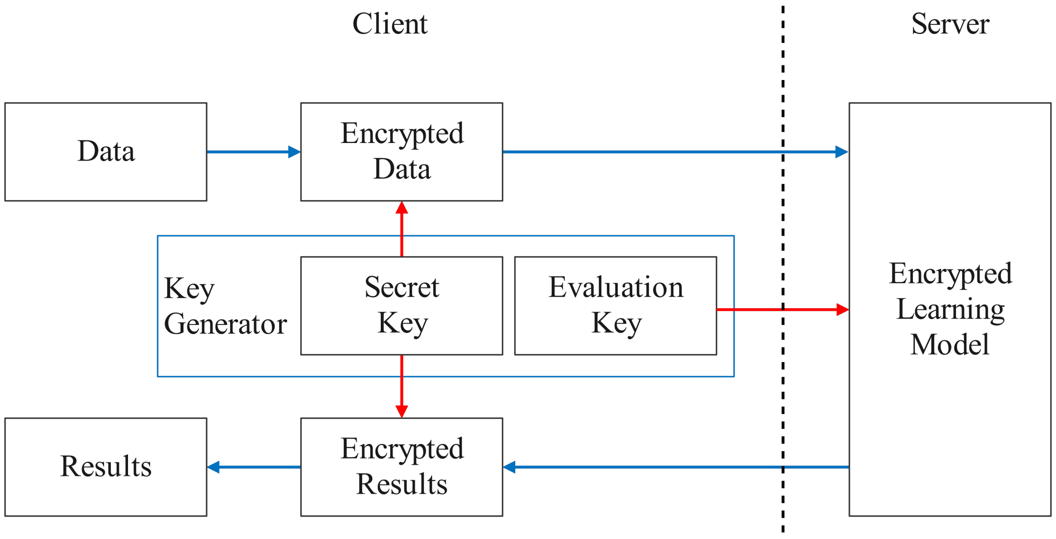

Figure 3. Privacy-preserving scheme for the learning model.

4.3.1. Privacy-Preserving Scheme Algorithms

We develop two algorithms that involve quantizing, encrypting, inferring, decrypting, and de-quantizing tasks. Algorithm 1 is designed for the client side, while Algorithm 2 is designed for the server side. Algorithm 1 starts with generating secret and evaluation keys through the KeyGen method. Then, the evaluation key is sent to the server through the SendKey method. Once the client receives the confirmation from the server, the input data are quantized (Quantization transforms values from a continuous domain $\mathbb{R}$ (Real numbers) to a discrete domain $\mathbb{I}$ (Integers)) through the QuantizeInput method and then the input data are encrypted through Encrypt method. Encrypted input is sent to the server to predict the output through MakeInference method. Finally, the client must wait until the server returns the encrypted output to decrypt and de-quantize it through Decrypt and DequantizeInput methods. As for Algorithm 2, it consists of two methods: SaveKey and MakeInference. SaveKey is responsible for storing the evaluation key sent from the client, while MakeInference predicts the output using the evaluation key and sends back the encrypted message containing the result to the client.

Algorithm 1 Tasks on the client side.

```
Require: Message $input$, Message $output$
Ensure: int riskLevel
 1: Initialize: boolean newKeys $\Leftarrow$ $True$, boolean received $\Leftarrow$ $False$, riskLevel $\Leftarrow$ 0;
 2: (secretKey, evalKey) $\Leftarrow$ KeyGen(newKeys);
 3: received $\Leftarrow$ SaveKey(evalKey);                    $\triangleright$ Waiting for server response
 4: quantized $\Leftarrow$ QuantizeInput(input);
 5: encrypted $\Leftarrow$ Encrypt(secretKey, quantized);                    $\triangleright$ It uses FHE
 6: if received = $True$ then
 7:     output $\Leftarrow$ MakeInference(encrypted);          $\triangleright$ Waiting for server response
 8:     if output $\neq$ $None$ then
 9:         decrypted $\Leftarrow$ Decrypt(secretKey, result);
10:         riskLevel $\Leftarrow$ DequantizeInput(decrypted);
11:     end if
12: end if
```

Algorithm 2 Tasks on the server side.

```
 1: Initialize: boolean evalKey $\Leftarrow$ $None$, int riskLevel = 0, Message output $\Leftarrow$ $None$;
 2: function SaveKey(key)
 3:     evalKey $\Leftarrow$ key;
 4:     return $True$
 5: end function
 6: function MakeInference(input)
 7:     if evalKey $\neq$ $None$ then
 8:         output $\Leftarrow$ Predict(evalKey, input);
 9:     end if
10:     return output
11: end function
```

4.3.2. Secrecy Proofs

The following lemmas and proofs demonstrate that our proposed scheme is secure under our Section 4.2.2. Thus, we declare four lemmas and their respective proofs.

Lemma 1. *Given a vehicular network $\mathcal{V}$, a cloud provider network $\mathcal{C}$, a set of authorized users of $\mathcal{V}$ ($\mathcal{U}_v$), a user $u \in \mathcal{U}_v$, a pair of messages $(m_i, m_o) \in \mathcal{M}$, a server $s \in \mathcal{C}$, and an adversary $\mathcal{A}$ such that $\mathcal{A} \in \mathcal{U}_v$ (Type I according to our Section 4.2.2), our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are IND-CPA secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{V}$.*

Proof of Lemma 1. Let sk_u be the secret key of u, ek_u be the evaluation key of u, and a pair of messages (m_i, m_o) such that $(m_i, m_o) \in \mathcal{M}$. First, u sends ek_u to s in C where the encrypted learning model is hosted (Algorithm 1 line 3 and Algorithm 2 line 2). Second, u encrypts m_i using sk_u and FHE (Algorithm 1 line 5). Third, u sends the encrypted message c_{m_i} to s. Fourth, s makes an inference using the encrypted model and returns the result in c_{m_o} to u (Algorithm 1 line 7 and Algorithm 2 line 6). It is assumed that $\mathcal{A}$ is located in $\mathcal{V}$ and $\mathcal{A}$ can listen to the communication channel. Finally, since $\mathcal{A} \in \mathcal{U}_v$, $\mathcal{A}$ intercepts (c_{m_i} or c_{m_o}) and ek_u and tries to decrypt c_{m_i} or c_{m_o}. However, $\mathcal{A}$ cannot decrypt c_{m_i} or c_{m_o} because sk_u is required (Algorithm 1 line 9). Given that the FHE algorithm is IND-CPA [35], our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{V}$. □

Lemma 2. *Given a vehicular network $\mathcal{V}$, a cloud provider network $\mathcal{C}$, a set of authorized users of $\mathcal{V}$ ($\mathcal{U}_v$), a user $u \in \mathcal{U}_v$, another vehicular network $\mathcal{Z}$, a set of authorized users of $\mathcal{Z}$ ($\mathcal{U}_z$), a pair of messages $(m_i, m_o) \in \mathcal{M}$, a server $s \in \mathcal{C}$, and an adversary $\mathcal{A}$ such that $\mathcal{A} \in \mathcal{U}_z$, but $\mathcal{A} \notin \mathcal{U}_v$ (Type*

II according to our Section 4.2.2), our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are IND-CPA secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{Z}$.

Proof of Lemma 2. Let sk_u be the secret key of u, ek_u be the evaluation key of u, and a pair of messages (m_i, m_o) such that $(m_i, m_o) \in \mathcal{M}$. First, u sends ek_u to s in $\mathcal{C}$, where the encrypted learning model is hosted (Algorithm 1 line 3 and Algorithm 2 line 2). Second, u encrypts m_i using sk_u and FHE (Algorithm 1 line 5). Third, u sends the encrypted message c_{m_i} to s. Fourth, s makes an inference using the encrypted model and returns the result in c_{m_o} to u (Algorithm 1 line 7 and Algorithm 2 line 6). It is assumed that $\mathcal{A}$ is located in $\mathcal{Z}$ and $\mathcal{A}$ perform any attack to gain privileges in $\mathcal{V}$. Finally, $\mathcal{A}$ intercepts c_{m_i} or c_{m_o} and ek_u and tries to decrypt c_{m_i} or c_{m_o}. However, $\mathcal{A}$ cannot decrypt c_{m_i} or c_{m_o} because sk_u is required (Algorithm 1 line 9). Given that the FHE algorithm is IND-CPA, our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{Z}$. □

Lemma 3. *Given a vehicular network $\mathcal{V}$, a cloud provider network $\mathcal{C}$, a set of authorized users of $\mathcal{V}$ ($\mathcal{U}_v$), a user $u \in \mathcal{U}_v$, the Internet network $\mathcal{I}$, a set of authorized users of $\mathcal{I}$ ($\mathcal{U}_i$), a pair of messages $(m_i , m_o) \in \mathcal{M}$, a server $s \in \mathcal{C}$, and an adversary $\mathcal{A}$ such that $\mathcal{A} \in \mathcal{U}_i$, but $\mathcal{A} \notin \mathcal{U}_v$ (Type III according to our Section 4.2.2), our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are IND-CPA secure even if $\mathcal{A}$ intercepts m_i or m_o in any place of $\mathcal{I}$.*

Proof of Lemma 3. Let sk_u be the secret key of u, ek_u be the evaluation key of u, and a pair of messages (m_i, m_o) such that $(m_i, m_o) \in \mathcal{M}$. First, u sends ek_u to s in $\mathcal{C}$ where the encrypted learning model is hosted (Algorithm 1 line 3 and Algorithm 2 line 2). Second, u encrypts m_i using sk_u and FHE (Algorithm 1 line 5). Third, u sends the encrypted message c_{m_i} to s. Fourth, s makes an inference using the encrypted model and returns the result in c_{m_o} to u (Algorithm 1 line 7 and Algorithm 2 line 6). It is assumed that $\mathcal{A}$ is located in any place of $\mathcal{I}$ and $\mathcal{A}$ perform any attack to intercept information of $\mathcal{V}$. Finally, $\mathcal{A}$ intercepts (c_{m_i} or c_{m_o}) and ek_u and tries to decrypt c_{m_i} or c_{m_o}. However, $\mathcal{A}$ cannot decrypt c_m because sk_u is required (Algorithm 1 line 9). Given that the FHE algorithm is IND-CPA, our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{I}$. □

Lemma 4. *Given a vehicular network $\mathcal{V}$, a set of authorized users of $\mathcal{V}$ ($\mathcal{U}_v$), a user $u \in \mathcal{U}_v$, a cloud provider network $\mathcal{C}$, a set of authorized users of $\mathcal{C}$ ($\mathcal{U}_c$), a pair of messages $(m_i, m_o) \in \mathcal{M}$, a server s $\in \mathcal{C}$, and an adversary $\mathcal{A}$ such that $\mathcal{A} \in \mathcal{U}_c$, but $\mathcal{A} \notin \mathcal{U}_v$ (Type IV according to our Section 4.2.2), our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are IND-CPA secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{C}$.*

Proof of Lemma 4. Let sk_u be the secret key of u, ek_u be the evaluation key of u, and a pair of messages (m_i, m_o) such that $(m_i, m_o) \in \mathcal{M}$. First, u sends ek_u to s in $\mathcal{C}$, where the encrypted learning model is hosted (Algorithm 1 line 3 and Algorithm 2 line 2). Second, u encrypts m_i using sk_u and FHE (Algorithm 1 line 5). Third, u sends the encrypted message c_{m_i} to s. Fourth, s makes an inference using the encrypted model and returns the result in c_{m_o} to u (Algorithm 1 line 7 and Algorithm 2 line 6). It is assumed that $\mathcal{A}$ is located in $\mathcal{C}$ and $\mathcal{A}$ can listen to the communication channel. Finally, since $\mathcal{A} \in \mathcal{U}_c$, $\mathcal{A}$ intercepts (c_{m_i} or c_{m_o}) and ek_u and tries to decrypt c_{m_i} or c_{m_o}. However, $\mathcal{A}$ cannot decrypt c_{m_i} or c_{m_o} because sk_u is required (Algorithm 1 line 9). Given that the FHE algorithm is IND-CPA, our algorithm guarantees that m_i sent by u from $\mathcal{V}$ to $\mathcal{C}$ or m_o sent by s from $\mathcal{C}$ to $\mathcal{V}$ are secure even if $\mathcal{A}$ intercepts m_i or m_o in $\mathcal{C}$. □

4.3.3. Complexity Analysis

In this subsection, we analyze the complexity of our proposal.

Data Preprocessing

We preprocessed the data by performing tasks such as filtering relevant columns. This process is performed in polynomial time, i.e., $\mathcal{O}(n)$, where n is the number of elements or size of the input. There is no significant computational overhead at this stage.

FHE Complexity

Since our security approach uses FHE, one has to analyze the complexity of the following steps: Key Generation (Algorithm 1-line 2), Compiling (Algorithm 1-line 4), Encryption 1-line 5, and Inference (Algorithm 2-line 6).

- Key Generation: For Homomorphic Encryption, the key generation process involves a polynomial operation depending on the security level (number of bits) [36]. Then, the complexity of this step is $\mathcal{O}(b)$, where b is the number of bits used according to the security level.
- Compiling: The compilation process in FHE refers to preparing a model (such as a decision tree or neural network) to be used in encrypted form, specifically transforming it into a format where computations can be performed homomorphically [37]. The model compilation phase involves operations like quantization, ciphertext packing, and transformation of operations into homomorphic equivalents. This compilation task has a quadratic time complexity as encryption is applied to each element [36]. Then, the complexity of this step is $\mathcal{O}(n^2)$, where n is the number of elements or size of the input.
- Encryption: We use a privacy-preserving machine learning framework to implement our scheme (see Section 5.2.2). This framework uses TFHE (Torus Fully Homomorphic Encryption) [36] or similar lattice-based encryption schemes. The complexity of encrypting a message in TFHE is typically linear in the size of the plaintext (which is impacted by the quantization bit-width), and polynomial in the security parameter [37]. Then, for this framework, the encryption complexity is $\mathcal{O}(n \times \lambda)$, where n is the number of bits in the quantized plaintext, i.e., size of the input (e.g., a 4-bit or 8-bit quantized value), and λ is the security parameter, which typically grows with the required level of encryption strength.
- Inference: In this framework, the complexity of FHE-based inference is the sum of the complexities of the individual operations required for the model and the complexity of the homomorphic operations. For decision trees, for instance, the complexity per prediction depends on the depth d of the tree, i.e., the complexity of the unencrypted model is $\mathcal{O}(d)$. Next, in the encrypted model, since each comparison and decision involves homomorphic operations, the complexity of a single prediction is $\mathcal{O}(d \times \lambda^2)$, where d is the depth of the decision tree and λ^2 represents the cost of each homomorphic operation. That is, the complexity is quadratic in the security parameter (λ^2) [37] and linear in the depth d of the tree or the complexity model that is being used.

5. Experiments

In this section, we describe the design, configuration, and implementation of the experiments.

5.1. Design

We designed two sets of experiments, each one consisting of 16 experiments. Both use a real driving dataset described in the Section 5.2.3. The first set uses models based on Decision Trees (DT), Random Forest (RF), Extreme Gradient Boosting (XGBoost), Multilayer Perceptron (MLP), and their equivalent homomorphic models. The first set uses two types of model configurations (default and optimal), which can be encrypted or unencrypted. The second set uses models based only on RF. In total, we conduct 32 experiments as part of this work. Finally, we use cross-validation with three folds to obtain the optimal configurations for the models, and we split the data into 70% for training and 30% for testing to evaluate models.

5.2. Setup

5.2.1. Computer Specs and Software

We performed all the experiments on a computer with an AMD Ryzen 5 processor (Advanced Micro Devices, Inc., Santa Clara, CA, USA) (6 cores and 12 threads) at 2.10 GHz, 34 GB of RAM, and one GPU (Radeon Graphics card) (Advanced Micro Devices, Inc., Santa Clara, CA, USA) at 1.8 GHz. The computer, equipped with a Windows 11 operating system, required additional software such as Docker (4.23.0) [38], WSL (2) [39], Python (3.8.10) [40], and glibc (2.31) [41].

5.2.2. Specialized Libraries

Our scheme uses Concrete ML (1.16) [42], a PPML set of tools based on Fully Homomorphic Encryption, to convert the learning model to its FHE equivalent. There is a Docker container, which includes all the software to run Concrete ML. This library is also available via the packet manager for Python (pip) [43]. An installation guide is available on the Concrete ML website. Its current version (1.6) supports the Python versions 3.8, 3.9, and 3.10. Our scheme also uses pandas (1.4.4) [44] for data analysis and manipulation, scikit-learn (1.1.13) [45] for building machine learning and neural network models, and pytorch (1.13.1) [46] for deep learning using graphics processing units (GPU).

5.2.3. Datasets

We used the public-access driving dataset POLIDriving [47] to train and test the learning models. POLIDriving contains more than 61 K observations, 32 attributes from five heterogeneous sources, and four classes (low, medium, high, and very high) representing the levels of risk of suffering a traffic accident. It includes time, speed, rpm, throttle position, engine temperature, engine load value, heart rate, current weather, visibility, precipitation, accidents on site, design speed, accidents by time, and others. For the experiments, we selected 1980 labeled observations and 14 attributes.

5.2.4. Models

Considering the algorithms/models used in related works (Table 1) and the available built-in models on Concrete ML, we considered tree-based models and neural networks. Thus, we chose Decision Tree (DT), Random Forest (RF), and Extreme Gradient Boosting (XGB) among tree-based models and Fully Connected (FCNN) among neural networks.

5.3. Configurations

The best configurations for the learning models were found after performing hyperparameter tuning. The hyperparameters and values for the learning models are presented as follows (Table 3). We provide four configurations for each model. The first two refer to the default and the optimal configuration for the unencrypted model, and the other two for the encrypted model. Some configurations include hyperparameters that are only applied to encrypted models.

Table 3. Configurations for encrypted and unencrypted learning models.

#	Algorithm	Configuration	Hyperparameters
1	Decision Tree	Default	criterion=‘gini’, max_depth=None, min_samples_leaf=1, min_samples_split=2, splitter=‘best’, random_state=42, *n_bits=6
2		Optimal	criterion: ‘gini’, max_depth=15, min_samples_leaf=1, min_samples_split=2, splitter=‘best’, random_state=42, *n_bits=5

Table 3. *Cont.*

#	Algorithm	Configuration	Hyperparameters
3	Random Forest	Default	criterion='gini', max_depth=4, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=20, random_state=42, n_jobs=1, *n_bits=6
4		Optimal	criterion='entropy', max_depth=15, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=10, random_state=42, n_jobs=1, *n_bits=4
5	XGBoost	Default	learning_rate=None, max_depth=3, n_estimators=20, random_state=42, n_jobs=1, *n_bits=6
6		Optimal	learning_rate=0.1, max_depth=15, n_estimators=10, random_state=42, n_jobs=1, *n_bits=5
7	FCNN	Default	hidden_layers=1, activation_function=None, criterion='entropy', optimizer='adam', learning_rate=0.01, max_epochs=10
8		Optimal	hidden_layers=2, activation_function='relu', criterion='entropy', optimizer='adam', learning_rate=0.01, max_epochs=100, *n_w_bits=4, *n_a_bits=4

* Only for encrypted models. criterion refers to the quality of a split. max_epochs is the maximum # of iterations. random_state control the randomness of data or splitting. n_jobs is the # of jobs to run in parallel. n_bits is the # of bits for quantization.

5.4. *Key Performance Metrics*

We use the metrics shown in Table 4 to evaluate the unencrypted and encrypted models.

Table 4. Key performance metrics.

Metric	Description	
	Unencrypted Model	**Encrypted Model**
Training time	It refers to the time it takes to train the learning model.	It refers to the time it takes to train the learning model using unencrypted data.
Compiling time	Not applicable.	It refers to the time it takes to convert a model into a code machine that executes the same model over encrypted data.
Key generation time	Not applicable.	It refers to the time spent generating the keys for encrypting and decrypting information.
Inference time	It refers to the time it takes to perform the predicting task.	It refers to the time it takes to perform quantizing, encrypting, predicting, decrypting, and dequantizing tasks.
Accuracy	It refers to the number of correct predictions about the total number of predictions.	It refers to the number of correct predictions about the total number of predictions.

6. Results

We evaluated the models using the metrics described in Section 5.4. Thus, Table 5 presents the results of the experiments using the POLIDriving dataset, and Table 6 presents the results of evaluating a Random Forest-based model using different configurations.

Table 5. Experimental results using the POLIDriving dataset.

Experiment	Algorithm	Encrypted	Configuration	Training Time	Compiling Time	Key Gen. Time	Inference Time	Accuracy
				[s]	[s]	[s]	[s/Sample]	
1	Decision Tree	No	Default	0.157	-	-	0.000078	82.0
2			Optimal	0.136	-	-	0.000056	85.9
3		Yes	Default	0.126	2.440	0.784	17.959156	82.0
4			Optimal	0.126	2.482	0.744	18.139494	85.9
5	Random Forest	No	Default	0.136	-	-	0.000016	69.7
6			Optimal	0.835	-	-	0.000672	85.0
7		Yes	Default	0.133	2.345	0.867	12.965843	69.7
8			Optimal	0.990	8.496	0.758	181.197287	85.0
9	XGBoost	No	Default	0.274	-	-	0.000029	81.5
10			Optimal	0.983	-	-	0.000622	84.8
11		Yes	Default	0.218	2.413	0.853	24.494245	81.5
12			Optimal	1.249	7.523	0.644	207.799554	84.8
13	Fully Connected Neural Network	No	Default	1.255	-	-	0.000015	71.0
14			Optimal	11.860	-	-	0.000041	80.1
15		Yes	Default	1.243	1.928	1.391	2.484339	71.0
16			Optimal	13.033	3.615	3.966	8.476462	80.1

Both unencrypted and encrypted models are trained using unencrypted data, resulting in no significant difference in training time between them. For instance, the decision tree-based model (Table 5, Experiments 2 and 4) reached training times of 0.136 and 0.126 s, the Random Forest-based model (Expts. 6 and 8) reached training times of 0.835 and 0.990 s, the XGBoost-based model (Expts. 10 and 12) reached training times of 0.983 and 1.249 s, and the FCNN-based model (Expts. 14 and 16) reached training times of 11.860 and 13.033 s According to Concrete ML, model training includes a quantization task performed after or before the training itself, depending on the model type. In that way, the models generated by Concrete ML experience higher training times than scikit-learn models.

Concerning compiling times, the decision tree-based model (Table 5, Expts. 3 and 4) obtained times of 2.440 s and 2.482 s using a default and optimal configuration, respectively. Compiling times are similar in this case because of the similarity of the configurations. For the other models that use optimal configurations, the compiling times are higher than models using default configurations. For instance, the Random Forest-based model (Expts. 7 and 8) obtained times of 2.345 and 8.496 s, the XGBoost-based model (Expts. 11 and 12) obtained times of 2.413 and 7.523 s, and the FCNN-based model (Expts. 15 and 16) obtained 1.928 and 3.615 s.

The key generation times for the tree-based models are similar (>0.6 and <0.9 s). Those models reached values of 0.784 (Table 5, Expt. 3), 0.744 (Expt. 4), 0.867 (Expt. 7), 0.758 (Expt. 8), 0.853 (Expt. 11), and 0.644 s (Expt. 12) using default and optimal configurations. In the case of the neural network-based models, there is a considerable increase compared to the results of the previous models. These last models reached values of 1.391 (Expt. 15) and 3.966 s (Expt. 16).

As expected, inference times are much higher in encrypted models than unencrypted ones. The results show that these differences are extremely high for some experiments, especially for the Random Forest and XGBoost-based models. For instance, Decision Tree-based models obtained times of 0.000078 (Table 5, Expt. 1) and 17.959156 s (Expt. 3) for unencrypted and encrypted models, respectively. However, the XGBoost-based models reached times of 0.000622 (Expt. 10) and 207.799 s (Expt. 12). The best results were obtained from the FCNN-based models, which reached values of 0.000041 (Expt. 14) and 8.476 s (Expt. 16).

Concerning the accuracy of the models, the results show no difference between encrypted and unencrypted ones. For instance, the Decision Tree-based models show an accuracy of 82.0 (Table 5, Expts. 1 and 3) for the unencrypted and encrypted models; the Random Forest-based models show an accuracy of 85.0 (Expts. 6 and 8) for both models; and the XGBoost-based models show accuracies of 84.8 (Expts. 10 and 12) for the unencrypted and encrypted models. Figure 4 presents a comparative analysis of the encrypted models using optimal configurations.

Additionally, we performed experiments to verify how the complexity of encrypted models affects the evaluation metrics. We chose the model based on RF because the difference in accuracy between the model using the default configuration and the one using the optimal configuration is greater, so any trend in values can be identified more easily. We also chose two parameters for hyperparameter tuning: depth and estimators. The results of these experiments are shown in Table 6, and the analysis of these results and the trends of each of the evaluation metrics are shown in Figure 5. Since it is difficult to determine if a configuration makes the model more or less complex than another when both parameters are increasing, we grouped the experiments into different sets (Groups 1 to 4). In these sets, only one of both parameters increases, so it is easy to determine that the current configuration makes the model more complex than the previous one.

Table 6. Evaluation of an FHE-based Random Forest model using different configurations.

Experiment	Group	Parameters		Training Time	Compiling Time	Key Generation Time	Inference Time	Accuracy
		Depth	Estimators	[s]	[s]	[s]	[s]	
1	1	5	5	0.155	1.987	0.747	5.267986	68.5
2		5	10	0.159	2.116	0.711	10.328380	69.7
3		5	20	0.312	3.299	0.719	21.743675	73.9
4		5	30	0.300	2.498	0.778	34.756006	74.4
5	2	10	5	0.359	3.801	0.545	64.507635	79.6
6		10	10	0.542	5.642	0.501	132.744697	83.7
7		10	20	1.093	10.623	0.554	266.056506	85.7
8		10	30	1.793	14.107	0.521	410.044873	86.9
9	3	20	5	0.533	6.089	0.562	98.318843	81.0
10		20	10	1.255	9.475	0.533	187.311936	83.5
11		20	20	2.903	17.774	0.531	257.538898	85.2
12		20	30	2.992	21.486	0.642	697.009592	84.5
13	4	30	5	0.550	5.317	0.518	71.353805	81.0
14		30	10	1.026	10.651	0.559	149.668607	83.5
15		30	20	2.189	15.220	0.537	294.049088	85.2
16		30	30	3.072	22.392	0.653	805.031174	84.5

RandomForest (max_depth=depth, n_estimators=estimators, criterion='entropy', random_state=42, n_bits=4, n_jobs=1).

The results confirmed that the values of the metrics (training, compiling, and inference times) increase as the model gets more complex. For instance, Group 2 (Expts. 8–12) goes from 0.359 to 1.793 s for training time, from 3.801 to 14.107 s for compiling time, and from 64.507 to 410.044 s for inference time. In the case of key generation time, it is not affected largely by the hyperparameter tuning. Those times vary between 0.501 s and 0.778 s. As can be seen in Figure 5, key generation time remains stable for all the experiments.

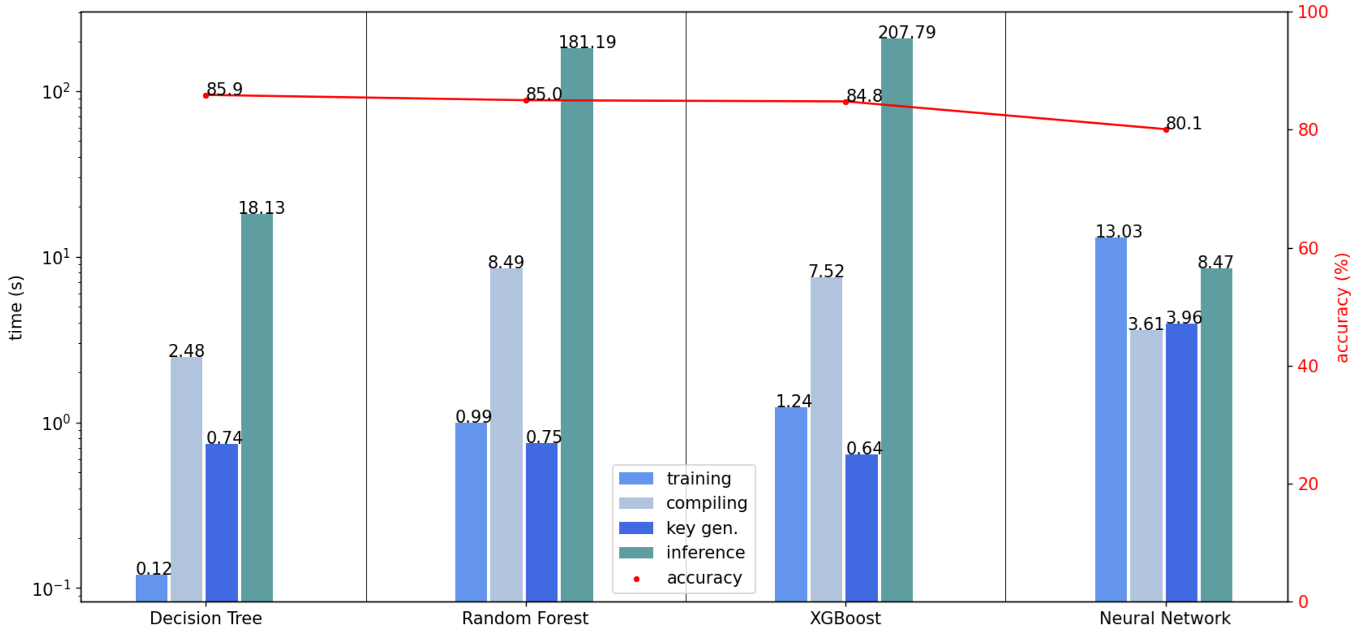

Figure 4. Analysis of the results obtained in evaluating the encrypted models using their optimal configurations (Table 5).

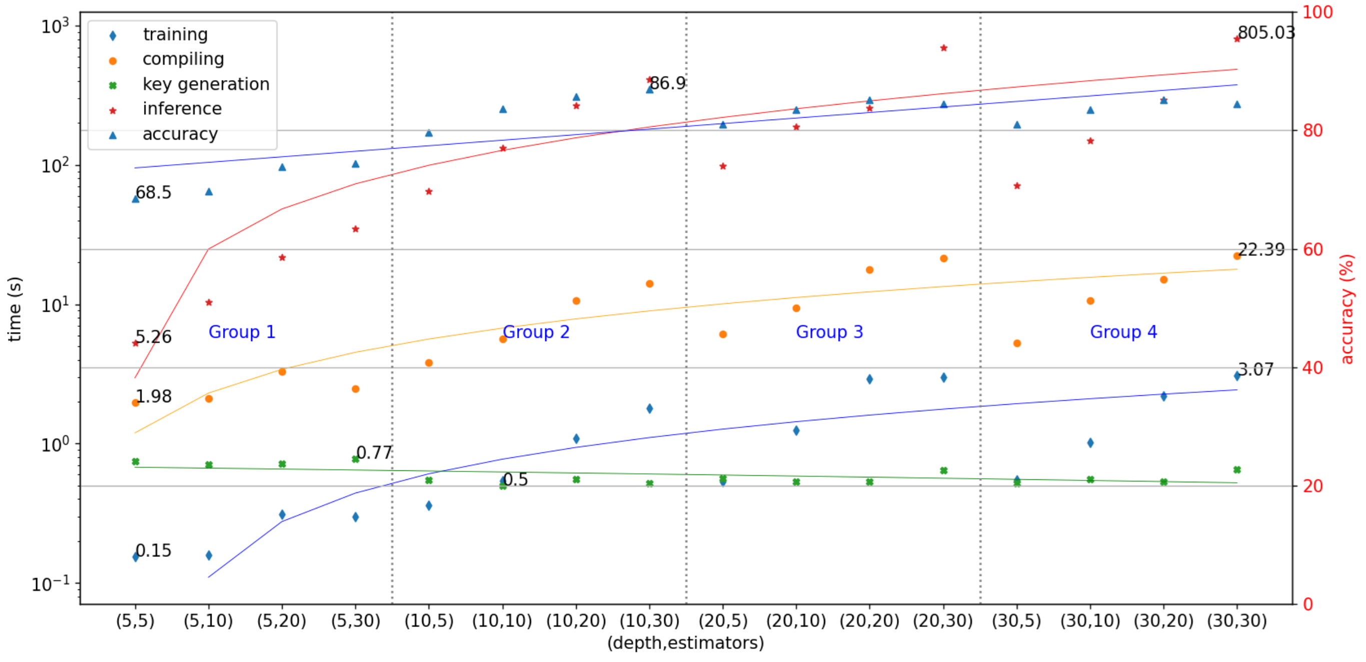

Figure 5. Analysis of the results obtained in evaluating the encrypted model based on Random Forest (Table 6).

Considering that the model used in Experiment 1 (Table 6) is less complex in terms of the number of depths and estimators and the one used in Experiment 8 is more complex, we identified an ascending trend in accuracy (see Figure 5). Thus, it increased from 68.5 (depth = 5, estimators = 5) to 86.9 (depth = 10, estimators = 30). The other metrics also suffered increases; some increases were huge. For instance, the compiling time increased from 1.987 (Expt. 1) to 14.107 s (Expt. 8), and the inference time increased from 5.267 to 410.044 s (see Figure 5).

7. Discussion

The inference times for encrypted models in all experiments are very high and unthinkable for real-time applications. In the first experiments, the results show a linear increase between inference time and model complexity; however, it turns to an exponential relation as model complexity keeps increasing. It is important to mention that the experiments were performed on a desktop computer with limited resources rather than on a high-performance server. Undoubtedly, these times will significantly decrease on a powerful server.

As expected, the unencrypted and encrypted models with the same configurations reach similar accuracy values. This result demonstrates the correct operation of Homomorphic Encryption in the learning models. Also, we noted a slight improvement in accuracy as hyperparameters change their values, which could be obvious as we are looking for the best configuration for the models; however, the inference time is greatly affected by tuning the models. This fact demonstrates that a complex model with better accuracy would affect the model performance by greatly increasing the inference time.

The time required to generate keys for tree-based models remains almost constant even if hyperparameters increase their values, except for neural network-based models, where the key generation time increases considerably. These results are expected since tree-based models are more computationally efficient than neural network-based models, which are more intensive. Similarly, training and compiling times vary as the hyperparameters vary. Thus, training time varies smoothly, and compiling time varies greatly.

Considering all the issues discussed previously, ideally, the aim would be to build a model that is so complex that its inference time is relatively low. In other words, there must be a balance between model complexity and model performance regarding inference time. Thus, the challenge is finding and tuning a model so that encryption, inference, and decryption times are low and its accuracy high.

The proposed system performs well with smaller datasets, but its performance depends on the hardware, particularly for the Fully Homomorphic Encryption (FHE) operations. As explained in the complexity analysis (Section 4.3.3), the system's complexity increases with larger datasets. However, the system's scalability can be improved in cloud-based environments by leveraging parallel processing and distributed computing. Each critical stage—key generation, compilation, encryption, and inference—can be optimized using cloud resources. While Homomorphic Encryption introduces significant overhead, especially in terms of computation and encryption complexity, modern cloud infrastructure allows these processes to be scaled across multiple nodes, improving efficiency without compromising security. Furthermore, the system can be optimized by balancing the security parameter λ against the desired performance, making it viable for large-scale applications. Another alternative is the nascent field of Federated Learning, in which a decentralized and distributed training process is performed to avoid sharing private data with a centralized server and sharing the computational cost among the parties.

Among all encrypted models, we noted that the model based on FCNN with the optimal configuration presents the best results, a balance between performance and efficiency. Additionally, we considered that although training and compilation times are high, these processes are performed only once during the model deployment. This model reached the highest accuracy for the lowest inference time (see Figure 4). Considering only the accuracy of the models, we confirmed that all other models obtained better accuracy values than the model based on FCNN. As discussed previously, the high computational cost can be resolved without problem using high-performance computational infrastructure. Therefore, the models based on Decision Trees, Random Forests, and XGBoost should not be discarded but should be contemplated in future analyses where this limitation has been resolved. Generally, high-performance servers run real-time applications, so deploying a real-time risk-level prediction system that counts with an additional security layer is fully viable in terms of performance and efficiency.

8. Conclusions and Future Work

We designed a privacy-preserving scheme for a traffic accident risk level prediction system that guarantees information confidentiality by making it unreadable for adversaries who can capture that information through an attack. This scheme is based on Fully Homomorphic Encryption (FHE). It guarantees that incoming and outgoing data of the prediction system are IND-CPA secure even if an attacker intercepts its data. Considering the adversary model, our privacy-preserving scheme protects the system from adversaries located in vehicular and cloud service provider networks and the Internet, as established in secrecy lemmas and proofs.

The high computational cost of performing operations over encrypted data affected the results, especially those related to inference time, which was as expected in line with the Section 4.3.3. These results allowed us to determine the weaknesses and challenges of implementing FHE. Despite these limitations, it is entirely feasible to implement FHE over learning models. Implementing FHE on a high-performance server will dramatically decrease the value of the key performance metrics. This scenario will even permit looking for the model configuration that offers the best performance without worrying about efficiency (shortest times).

Future work must include evaluating the privacy-preserving scheme using other specialized libraries (e.g., Seal and OpenFHE). Additionally, this evaluation must be performed in a production environment provided by a cloud service provider to confirm the real values of the evaluation metrics. This evaluation will provide information to determine how far the complexity of the learning model, at the expense of efficiency, is relevant. In other words, if it is necessary to sacrifice the complexity of the learning model by getting low response times.

Finally, we plan to implement federated learning on cloud service providers (e.g., Amazon Web Service and Google Clouds) as an alternative for preserving data privacy. The idea is that data vehicle and driver's data, which include sensitive information such as vehicle and driver's ID and location, are processed into the acquisition modules. Information from the other data sources (weather conditions, traffic accidents, and road geometric characteristics), which do not include sensistive information and are public access, are processed from the centralized server.

Author Contributions: Conceptualization, P.M. and G.S.; methodology, P.M. and G.S.; investigation, P.M.; writing—original draft preparation, P.M.; writing—review and editing, P.M., G.S. and M.H.-Á.; supervision, G.S. and M.H.-Á. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: Our recognition to VIIV (Vicerrectorado de Investigación, Innovación y Vinculación) of Escuela Politécnica Nacional.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Marcillo, P.; Valdivieso Caraguay, Á.L.; Hernández-Álvarez, M. A Systematic Literature Review of Learning-Based Traffic Accident Prediction Models Based on Heterogeneous Sources. *Appl. Sci.* **2022**, *12*, 4529. [CrossRef]
2. Yuan, Z.; Zhou, X.; Yang, T. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 984–992.
3. Huang, T.; Wang, S.; Sharma, A. Highway crash detection and risk estimation using deep learning. *Accid. Anal. Prev.* **2020**, *135*, 105392. [CrossRef] [PubMed]

4. Basso, F.; Basso, L.J.; Bravo, F.; Pezoa, R. Real-time crash prediction in an urban expressway using disaggregated data. *Transp. Res. Part C Emerg. Technol.* **2018**, *86*, 202–219. [CrossRef]
5. Glavić, D.; Mladenović, M.; Stevanovic, A.; Tubić, V.; Milenković, M.; Vidas, M. Contribution to accident prediction models development for rural two-lane roads in Serbia. *Promet-Traffic Transp.* **2016**, *28*, 415–424. [CrossRef]
6. Kodepogu, K.; Manjeti, V.; Siriki, A. Machine learning for road accident severity prediction. *Mechatron. Intell. Transp. Syst.* **2023**, *2*, 211–226. [CrossRef]
7. Vizitiu, A.; Nitja, C.I.; Puiu, A.; Suciu, C.; Itu, L.M. Applying deep neural networks over homomorphic encrypted medical data. *Comput. Math. Methods Med.* **2020**, *2020*, 3910250. [CrossRef]
8. Bajard, J.C.; Martins, P.; Sousa, L.; Zucca, V. Improving the efficiency of SVM classification with FHE. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1709–1722. [CrossRef]
9. Li, J.; Kuang, X.; Lin, S.; Ma, X.; Tang, Y. Privacy preservation for machine learning training and classification based on homomorphic encryption schemes. *Inf. Sci.* **2020**, *526*, 166–179. [CrossRef]
10. Hong, S.; Park, J.H.; Cho, W.; Choe, H.; Cheon, J.H. Secure tumor classification by shallow neural network using homomorphic encryption. *BMC Genom.* **2022**, *23*, 284 . [CrossRef]
11. Kim, A.; Song, Y.; Kim, M.; Lee, K.; Cheon, J.H. Logistic regression model training based on the approximate homomorphic encryption. *BMC Med. Genom.* **2018**, *11*, 23–31. [CrossRef]
12. Kim, M.; Song, Y.; Wang, S.; Xia, Y.; Jiang, X. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR Med. Inform.* **2018**, *6*, e8805. [CrossRef] [PubMed]
13. Popescu, A.B.; Taca, I.A.; Nita, C.I.; Vizitiu, A.; Demeter, R.; Suciu, C.; Itu, L.M. Privacy preserving classification of eeg data using machine learning and homomorphic encryption. *Appl. Sci.* **2021**, *11*, 7360. [CrossRef]
14. Marcano, N.J.H.; Moller, M.; Hansen, S.; Jacobsen, R.H. On fully homomorphic encryption for privacy-preserving deep learning. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
15. Wu, W.; Liu, J.; Wang, H.; Hao, J.; Xian, M. Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique. *IEEE Trans. Knowl. Data Eng.* **2020**, *33*, 3424–3437. [CrossRef]
16. Sun, X.; Zhang, P.; Liu, J.K.; Yu, J.; Xie, W. Private machine learning classification based on fully homomorphic encryption. *IEEE Trans. Emerg. Top. Comput.* **2018**, *8*, 352–364. [CrossRef]
17. Lee, J.W.; Kang, H.; Lee, Y.; Choi, W.; Eom, J.; Deryabin, M.; Lee, E.; Lee, J.; Yoo, D.; Kim, Y.S.; et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access* **2022**, *10*, 30039–30054. [CrossRef]
18. Cheon, J.H.; Kim, D.; Kim, Y.; Song, Y. Ensemble method for privacy-preserving logistic regression based on homomorphic encryption. *IEEE Access* **2018**, *6*, 46938–46948. [CrossRef]
19. Han, B.; Kim, Y.; Choi, J.; Shin, H.; Lee, Y. Fully homomorphic privacy-preserving naive Bayes machine learning and classification. In Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Copenhagen, Denmark, 26 November 2023; pp. 91–102.
20. Sarkar, E.; Chielle, E.; Gürsoy, G.; Mazonka, O.; Gerstein, M.; Maniatakos, M. Fast and scalable private genotype imputation using machine learning and partially homomorphic encryption. *IEEE Access* **2021**, *9*, 93097–93110. [CrossRef]
21. Park, S.; Byun, J.; Lee, J.; Cheon, J.H.; Lee, J. HE-friendly algorithm for privacy-preserving SVM training. *IEEE Access* **2020**, *8*, 57414–57425. [CrossRef]
22. Han, K.; Hong, S.; Cheon, J.H.; Park, D. Logistic regression on homomorphic encrypted data at scale. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 9466–9471.
23. Yu, S.; Lee, J.; Lee, K.; Park, K.; Park, Y. Secure authentication protocol for wireless sensor networks in vehicular communications. *Sensors* **2018**, *18*, 3191. [CrossRef]
24. Syed, T.A.; Siddique, M.S.; Nadeem, A.; Alzahrani, A.; Jan, S.; Khattak, M.A.K. A novel blockchain-based framework for vehicle life cycle tracking: An end-to-end solution. *IEEE Access* **2020**, *8*, 111042–111063. [CrossRef]
25. Liu, Y.; James, J.; Kang, J.; Niyato, D.; Zhang, S. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet Things J.* **2020**, *7*, 7751–7763. [CrossRef]
26. Kim, M. HELR. Available online: https://github.com/K-miran/HELR (accessed on 1 June 2024).
27. Microsoft. SEAL. Available online: https://github.com/microsoft/SEAL (accessed on 1 June 2024).
28. Halevi, S. HElib. Available online: https://github.com/homenc/HElib (accessed on 1 June 2024).
29. Kim, A. HEAAN. Available online: https://github.com/kimandrik/HEAAN (accessed on 1 June 2024).
30. Podschwadt, R.; Takabi, D.; Hu, P.; Rafiei, M.H.; Cai, Z. A survey of deep learning architectures for privacy-preserving machine learning with fully homomorphic encryption. *IEEE Access* **2022**, *10*, 117477–117500. [CrossRef]
31. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
32. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv. (Csur)* **2018**, *51*, 1–35. [CrossRef]
33. Faust, S.; Masny, D.; Venturi, D. Chosen-ciphertext security from subset sum. In *Public-Key Cryptography–PKC 2016, Proceedings of the 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, 6–9 March 2016*; Proceedings, Part I; Springer: Berlin/Heidelberg, Germany, 2016; pp. 35–46.

34. Marcillo, P.; Tamayo-Urgilés, D.; Valdivieso Caraguay, Á.L.; Hernández-Álvarez, M. Security in V2I Communications: A Systematic Literature Review. *Sensors* **2022**, *22*, 9123. [CrossRef]
35. Fauzi, P.; Hovd, M.N.; Raddum, H. On the IND-CCA1 security of FHE schemes. *Cryptography* **2022**, *6*, 13. [CrossRef]
36. Frery, J.; Stoian, A.; Bredehoft, R.; Montero, L.; Kherfallah, C.; Chevallier-Mames, B.; Meyre, A. Privacy-preserving tree-based inference with fully homomorphic encryption. *Cryptology ePrint Archive* **2023**.
37. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachène, M. TFHE: fast fully homomorphic encryption over the torus. *J. Cryptol.* **2020**, *33*, 34–91. [CrossRef]
38. Docker Inc. Docker. Available online: https://www.docker.com/ (accessed on 1 June 2024).
39. Microsoft. WSL. Available online: https://learn.microsoft.com/en-us/windows/wsl/ (accessed on 1 June 2024).
40. Python Software Foundation. Python. Available online: https://www.python.org/ (accessed on 1 June 2024).
41. GNU Project. The GNU C Library. Available online: https://www.gnu.org/software/libc/ (accessed on 1 June 2024).
42. Zama. Concrete ML. Available online: https://docs.zama.ai/concrete-ml/ (accessed on 1 June 2024).
43. Bicking, I. The packet installer for Python. Available online: https://pypi.org/project/pip/ (accessed on 1 June 2024).
44. McKinney, W. Powerful Python data analysis toolkit. Available online: https://pypi.org/project/pandas/ (accessed on 1 June 2024).
45. Scikit-learn Developers. Scikit-learn. Available online: https://scikit-learn.org/ (accessed on 1 June 2024).
46. The Linux Foundation. Pytorch. Available online: https://pypi.org/project/torch/ (accessed on 1 June 2024).
47. Marcillo, P.; Arciniegas-Ayala, C.; Valdivieso Caraguay, Á.L.; Sanchez-Gordon, S.; Hernández-Álvarez, M. POLIDriving: A Public-Access Driving Dataset for Road Traffic Safety Analysis. *Appl. Sci.* **2024**, *14*, 6300. [CrossRef]