EVAD: Encrypted Vibrational Anomaly Detection With Homomorphic Encryption

Alessandro Falcetta[1*] and Manuel Roveri[1]

[1*]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, Milano, 20133, MI, Italy.

*Corresponding author(s). E-mail(s): alessandro.falcetta@polimi.it;
Contributing authors: manuel.roveri@polimi.it;

Abstract

One of the main concerns of cloud-based services based on machine and deep learning algorithms is the privacy of users' data. This is particularly relevant when companies want to leverage such services because they have to outsource potentially sensible data to be processed. In this work, the problem of privacy-preserving anomaly detection on industrial vibrational data with machine learning is tackled. It consists in the detection of irregularities or deviations from expected patterns in the vibration signals generated by industrial machinery and equipment. Such anomalies can be indicative of potential equipment failures, maintenance needs, or process deviations, making their timely detection critical for ensuring the smooth operation and reliability of industrial systems. We combine this industrial need with the ability to guarantee data privacy by proposing Encrypted Vibrational Anomaly Detection (EVAD). EVAD allows the detection of anomalies on vibrational data in a privacy-preserving manner by integrating, for the first time in the literature, One-Class Support Vector Machines (OCSVM) and Homomorphic Encryption (HE), the latter being a particular kind of encryption that allows the computation of some operations directly on encrypted data. Experimental results show that, on two publicly available datasets for vibrational anomaly detection, EVAD is able to distinguish, in a privacy-preserving manner, between nominal and anomaly situations, in an effective and efficient way. To the best of our knowledge, EVAD represents the first privacy-preserving solution for the detection of anomalies in vibrational data present in the literature.

Keywords: homomorphic encryption, machine learning, anomaly detection, vibrational signals

1 Introduction

The rise of privacy-preserving Machine Learning (ML) and Deep Learning (DL) has opened a new era of computing solutions able to provide services based on ML and DL, meanwhile guaranteeing the privacy of users' data. In this perspective, being able to guarantee the privacy of people [27, 39] or industrial entities [6, 29] during the processing of data is one of the most promising and challenging research fields from the technical and ethical point of view [20]. The privacy-preserving processing of data is particularly relevant for industries and companies, since they are driven by concerns regarding competitive disadvantage or potential data breaches, hence refraining from employing cloud-based ML/DL solutions. This constitutes an entry barrier that hinders these companies from harnessing value from their own data [48].

Anomaly detection is an important technique for recognizing unusual patterns in data that do not conform to an expected behavior. Anomaly detection can be applied in a variety of domains [41] such as fraud detection [2], medical diagnosis [21], and fault detection [17]. Of particular interest for many industrial

entities is the task of anomaly detection on vibrational data (e.g., data coming from sensors installed on industrial machinery). One of the main advantages of applying anomaly detection in this field is that it enables predictive maintenance, which is known to reduce operating costs and improve the reliability of machinery [55]. The literature about anomaly detection on vibrational data is wide, and several works are available, e.g., [40, 53]. None of these works addresses the problem of privacy in anomaly detection.

In the field of privacy-preserving computation, among the various technologies available able to guarantee the privacy of users [11], Homomorphic Encryption (HE) [18] is receiving increasing attention from both the academic and industrial sectors [31]. By leveraging HE schemes, solutions based on ML and DL can execute operations on encrypted data without the need to decrypt them and ensuring the accuracy of the results. In other words, HE protects the data not only at rest and in transit [52], but also during the processing [13]. This is particularly useful in application scenarios such as Cloud-based services where data are provided by users to the Cloud for the processing in an as-a-service manner. However, HE introduces two main limitations: (i) the operations on encrypted data are characterized by a severe computational overhead with respect to the same operations performed on plain data, requiring 50x-100x more computational time and memory [15]; (ii) only some operations are permitted on encrypted data (i.e., the majority of HE schemes allow only additions and multiplications). This point can explain why the literature on privacy-preserving ML and DL solutions implementing HE is still limited (being ML and DL solutions characterized by a long sequence of non linear operations). The few existing solutions present in the literature mainly focus on privacy-preserving image and text classification [18], while HE is expected, in the next years, to become viable for many different real-world use cases [25].

The goal of this paper is to introduce Encrypted Vibrational Anomaly Detection (EVAD). EVAD is a solution able to detect anomalies on users' vibrational signals meanwhile respecting their privacy. This is done by leveraging HE and One-Class Support Vector Machines (OCSVM) [12, 46], which are an evolution of Support Vector Machines (SVM) [16], specifically designed for the task of anomaly detection. To the best of our knowledge, EVAD represents the first privacy-preserving solution for the detection of anomalies in vibrational data present in the literature. The innovative contributions of this study are:

1. The introduction of the EVAD pipeline, a privacy-by-design pipeline for vibrational anomaly detection. EVAD is designed to be deployed in an as-a-service manner;
2. The re-design of conventional algorithms for anomaly detection, to make it possible to process encrypted data. This involves both the data pre-processing part (i.e., the DFT transformation, which is done on encrypted inputs) and the inference of the model (i.e., the aggregation of the intermediate encrypted representations in a single vector and the inference algorithm of the OCSVM);
3. The use of both HE optimization techniques and a feature reduction mechanism, to reduce the computational demand of EVAD. This is done to comply with the requirements of an as-a-service solution.

All the codes used in this paper are released to the scientific community as a public repository.[1]

The paper is organized as follows. Section 2 describes the related literature and Section 3 describes the background on HE and OCSVMs. Section 4 introduces the proposed EVAD solution. Lastly, the experimental results are given in Section 5, and the conclusions are finally drawn in Section 6.

2 Related literature

The literature in the field of privacy-preserving ML and DL focuses in particular on the processing of images, for which it presents various examples of privacy-preserving solutions, leveraging in particular Convolutional Neural Networks [34] (CNNs). One of the first works in this field is [23], which introduced a 5-layer CNN able to make image classification on an encrypted dataset. Another example is [35], which introduced various optimizations in order to implement the ResNet-110 [26] CNN on encrypted images.

Other examples of application of privacy-preserving solutions with HE are natural language processing (e.g., [14, 54]), genomic data processing (e.g., [33]), and time-series forecasting (e.g., [19]).

Moving to the anomaly detection task considered in this study, very few works are available in the literature. In [4], the authors proposed a solution for

[1]The code is attached to this submission, and will be made publicly available in the final version of the paper.

the detection of anomalies in data coming from environmental sensors (e.g., temperature, humidity, etc.). However, the employed HE scheme does not follow the more recent standards in terms of security, and the procedure requires the active collaboration of the client to perform some intermediate computations. This is a limitation present also in [7], where the authors presented a control and anomaly detection pipeline for industrial plants. In this work, the detection of anomalies is performed by computing a suitably defined statistic called nonparametric cumulative sum (CUSUM).

On the contrary, [5] proposed a solution based on a distributed version of the fuzzy c-means clustering algorithm (FCM) on encrypted data, applied on environmental data. Nonetheless, the implementation of a custom mechanism to process floating point values results in a very high computational cost, requiring a massive parallelization over 100 computational units to perform a single inference task in a reasonable amount of time. Differently, SigML [51] proposed a mechanism for supervised log anomaly detection, where a device sends encrypted logs to a third party, and logistic regression is used as ML model. The result is sent back to the device, where it will be decrypted. SigML leverages the same encryption scheme used in this work, i.e., the Cheon-Kim-Kim-Song (CKKS) scheme, which will be detailed in Section 3. Another example of anomaly detection on encrypted data is given in [49], where a method for encrypted intrusion detection is proposed. A client can send encrypted raw data (such as metrics), and obtain an encrypted alert that can be decrypted and triggered if an intrusion is detected. While potentially of interest, the mechanisms used for anomaly detection in this case are specific to the application scenario of intrusion detection.

For what concerns the use of the SVM models family, three works deserve attention. In [42] the authors proposed a method which supports fair learning of SVMs on encrypted data, using HE. In particular, they aimed to protect the privacy of some sensitive variables that could be present in datasets containing personal information (e.g., income). Moreover, a regulator, which is another third-party which mediates the processing between the client and the entity which trains the model, has to be involved. Another work which uses SVM combined with HE is introduced in [28], in which a privacy-preserving scheme for image classification is provided. In that case, the feature extraction part (which could be compared with the first steps of the EVAD pipeline, see Section 4.1) is done on the client, before the encryption. Lastly, [3] proposed a SVM algorithm on encrypted data for the classification of Bitcoin transactions. This work shares many similarities with the paper at hand, including the use of the CKKS scheme. The differences lay in the considered task (i.e., classification versus anomaly detection) and in the learning approach (i.e., supervised versus unsupervised). Moreover, the preprocessing phase introduced in Section 4 allows us to process signals which require an analysis in the frequency domain.

3 Background

3.1 Homomorphic encryption

HE is a family of encryption schemes that support the computation of some operations directly over encrypted data [1]. In particular, an encryption scheme with an encryption function $E(\cdot)$ and decryption function $D(\cdot)$ is considered homomorphic with respect to a class of functions $\mathcal{F}$ if, for any $f \in \mathcal{F}$, a function g can be constructed such that

$$f(x) = D(g(E(x)))$$

where x is any set of possible inputs [10]. Among the available HE schemes, in this study we opted for the CKKS scheme [15], which is based on the *Ring Learning With Errors* (RLWE) problem [37]. The CKKS scheme is an *approximate* scheme, in the sense that it supports two operations (i.e., additions and multiplications) on encrypted real values, with a certain precision. Indeed, HE schemes inject noise into ciphertexts to guarantee the probabilistic encryption properties [24], and this has two consequences: first, every time an operation is performed on a CKKS ciphertext, the amount of noise in that ciphertexts increases. This will lead to a bigger rounding error when it is decrypted. Second, a limited number of consecutive multiplications on a CKKS ciphertext can be done before it gets corrupted. This number is often called *level* of the scheme, and is indicated with l. The scheme can be configured through a set of encryption parameters $\Theta = [N, q, \Delta]$, where N is a power of 2 and it is called polynomial modulus, $q = [q_0, \ldots, q_{L+1}]$ is the list of $l+2$ prime numbers, which are the coefficient modulus, and Δ is a power of 2 that is the precision factor. Θ defines the algebraic structure of the plaintexts and ciphertexts. In particular, plaintexts are in the

polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, while ciphertexts are in the polynomial ring $\mathcal{R}_q = \mathbb{Z}_{q_0}[X]/(X^N + 1)$. Each time a ciphertext is multiplied with another ciphertext or a plaintext, the ciphertext coefficient modulus is switched to the next one in the list q (e.g., a ciphertext having coefficient modulus q_0 will have coefficient modulus q_1 after a multiplication). This can be done up to using q_L, after which no more multiplications are permitted. Δ is the precision of the encoding of freshly encrypted real numbers. There are three key elements that should be considered when dealing with the CKKS scheme.

The *first* is the encryption parameters choice. In particular, the choice of Θ affects the level of the scheme l, the computational overhead of the operations with respect to the same ones applied on plaintexts, and the security level. For these reasons, it is suggested to rely on state-of-the-art software libraries, which provide helper functions to tune Θ. In particular, in this work the SEAL library [47] is used, through the Python wrapper TenSEAL [8]. SEAL requires the user to specify l and b, which is the list of bit-lengths of the numbers in q. From these values, a suitable value for N is selected. Moreover, SEAL will securely compute the actual values for the coefficient modulus q, which will have the bit-lengths specified in b. More in detail, a possible choice for b is $b = [60, b_1, \ldots, b_L, 60]$, where b_1 and b_L should be equal to the exponent of the desired precision factor Δ. As an example, if the desired precision factor is $\Delta = 2^{50}$, and $l = 3$ multiplications have to be done on a ciphertext, a possible choice for b is $b = [60, 50, 50, 50, 60]$. The sum of the values in b defines the minimum possible available value for N, given that the choice of N defines an upper limit on this sum to maintain the scheme secure. It is suggested to use the smallest possible N, because increasing N increases the dimension of both plaintexts and ciphertexts, which results in heavier operations. Such thresholds are defined to guarantee the security of the scheme.

The *second* one is the concept of batching. To encrypt real values, it is first necessary to encode them into proper polynomial plaintexts. The CKKS scheme provides a mechanism to encode a vector $\mathbf{z} \in \mathbb{C}^{N/2}$ of size $N/2$ into a polynomial $m(X) \in \mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$ by using a variant of the complex canonical embedding map $\phi : \mathbb{C}^{N/2} \rightarrow \mathcal{R}$ [38]. By using this encoding technique, a single plaintext (or, equivalently, a single ciphertext) can store up to $N/2$ complex values. The practice of storing multiple values into a single plaintext or ciphertext is defined *batching* [50]. Batching enables the application of parallel processing through *Single Instruction, Multiple Data (SIMD)* operations [22]. This greatly reduces the computational burden of HE, both for the times and memory required for the operations.

The *third* one concerns the use of relinearization keys and Galois keys, two kinds of additional public keys. The relinearization key is needed to perform the relinearization operation, which is needed after a multiplication. Indeed, in CKKS a multiplication between two ciphertexts drastically increases the size of the resulting ciphertext. The relinearization operation allows to fix this problem by transforming back the ciphertext to its normal dimension. The Galois keys are needed to perform the rotation operation on batched ciphertexts. The rotation operation is needed to perform the dot-product between ciphertexts and plaintexts, as well as to perform the vector-matrix multiplications. These keys should be generated by the same entity which generates the public and secret keys pair, and should be outsourced to a third-party if the relinearization and rotation operations need to be computed by them. We stress that sharing these keys does not allow the third-party to decrypt ciphertexts in any case.

The CKKS scheme supports a number of operations on encrypted vectors. Let $\widehat{a} = [\widehat{a_0}, \widehat{a_1}, \ldots, \widehat{a_n}]$ and $\widehat{b} = [\widehat{b_0}, \widehat{b_1}, \ldots, \widehat{b_n}]$ be two encrypted CKKS vectors. The first type of operation natively available in CKKS is the element-wise addition between encrypted vectors:

$$\widehat{a} + \widehat{b} = [\widehat{a_0} + \widehat{b_0}, \widehat{a_1} + \widehat{b_1}, \ldots, \widehat{a_n} + \widehat{b_n}]. \tag{1}$$

The second type of operation is the element-wise multiplications between encrypted vectors:

$$\widehat{a}\widehat{b} = [\widehat{a_0}\widehat{b_0}, \widehat{a_1}\widehat{b_1}, \ldots, \widehat{a_n}\widehat{b_n}]. \tag{2}$$

While not being natively available in the CKKS scheme, two more operations used in this work can be implemented on top of other operations provided by the scheme: the dot-product between encrypted vectors and the matrix-matrix multiplication. The first is defined as:

$$\widehat{a} \cdot \widehat{b} = \widehat{a_0}\widehat{b_0} + \widehat{a_1}\widehat{b_1} + \ldots + \widehat{a_n}\widehat{b_n}. \tag{3}$$

While the latter is defined as follows:

$$\begin{bmatrix} \widehat{a}_{11} & \widehat{a}_{12} & \cdots & \widehat{a}_{1n} \\ \widehat{a}_{21} & \widehat{a}_{22} & \cdots & \widehat{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{a}_{m1} & \widehat{a}_{m2} & \cdots & \widehat{a}_{mn} \end{bmatrix} \begin{bmatrix} \widehat{b}_{11} & \widehat{b}_{12} & \cdots & \widehat{b}_{1p} \\ \widehat{b}_{21} & \widehat{b}_{22} & \cdots & \widehat{b}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{b}_{n1} & \widehat{b}_{n2} & \cdots & \widehat{b}_{np} \end{bmatrix} = \begin{bmatrix} \widehat{c}_{11} & \widehat{c}_{12} & \cdots & \widehat{c}_{1p} \\ \widehat{c}_{21} & \widehat{c}_{22} & \cdots & \widehat{c}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \widehat{c}_{m1} & \widehat{c}_{m2} & \cdots & \widehat{c}_{mp} \end{bmatrix} \tag{4}$$

where:

$$\widehat{c}_{ij} = \widehat{a}_{i1}\widehat{b}_{1j} + \widehat{a}_{i2}b_{2j} + \cdots + \widehat{a}_{in} + \widehat{b}_{nj} = \sum_{k=1}^{n} \widehat{a}_{ik}\widehat{b}_{kj}.$$

These operations will be used in the encrypted processing of EVAD. Their implementation is beyond the scope of this work, and more details can be found in [8].

3.2 One-Class Support Vector Machines

The One-Class Support Vector Machine (OCSVM) algorithm [12, 46] is an adaptation of the Support Vector Machine (SVM) algorithm, modified for the case of one-class classification. Unlike traditional classification problems, where the goal is to distinguish between two or more classes, OCSVM focuses on identifying instances being outliers or anomalies within a given dataset. This approach is particularly valuable in situations where the majority of the training data belong to a known class, and the objective is to uncover rare and unexpected instances.

OCSVMs, similarly to the SVM models family, leverage the concept of finding an optimal hyperplane to separate data points in a high-dimensional feature space. In the context of one-class classification, the primary goal is to define a boundary that encapsulates the normal data points while maximizing the margin between this boundary and the data. Instances falling outside this margin are considered anomalies.

An important hyperparameter used in the training of OCSVM is the parameter ν. This parameter controls the proportion of outliers the model is allowed to classify as nominal. In other words, ν defines the upper bound on the fraction of training data that can be considered as misclassifications or outliers. In the next Section the proposed solution will be introduced.

4 The proposed solution

Let $X \in \mathbb{R}^{M \times L}$ be a collection of M signals of length L (e.g., M is the number of axes in a vibrational dataset). The goal of the anomaly detection service proposed in this work is to compute $y = f_\theta(X)$, where $y \in \mathbb{R}$ is a real value indicating if X is nominal (i.e., its frequency spectrum is compatible with the one of the data the model $f_\theta(\cdot)$ was trained on, and y will be positive) or anomalous (i.e., its frequency spectrum data cannot be considered compatible with the one of the data the model $f_\theta(\cdot)$ was trained on, and y will be negative). We emphasize that the model $f_\theta(\cdot)$ is trained only on nominal data, i.e., data not containing any anomaly.

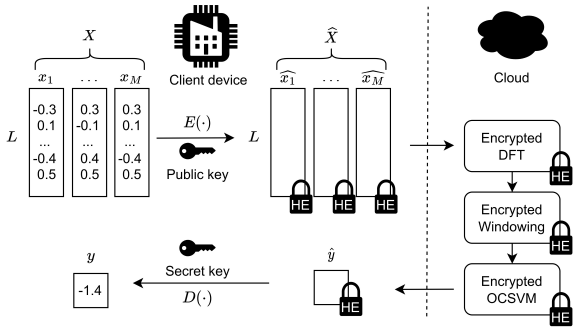

Fig. 1: The architecture of the proposed EVAD.

The peculiarity of the proposed EVAD solution is to provide such an anomaly detection service on encrypted data, hence respecting the privacy of clients' data.

An overview of the proposed EVAD solution is given in Fig. 1. The collection of signals $X = [x_1, \ldots, x_M]$ is encrypted with an encryption function $E(\cdot)$ and a public key on the client device, obtaining $\widehat{X} = [\widehat{x_1}, \ldots, \widehat{x_M}]$. $\widehat{X}$ is then sent to the third-party service, operating on the Cloud. Then, a pipeline of privacy-preserving transformations is applied to the encrypted data. This allows the third-party to process the client's signals $\widehat{X}$ in a privacy-preserving way, guaranteeing that both the data and the intermediate results remain encrypted during all the steps of the pipeline. The pipeline, detailed in Fig. 2, is organized into the following steps: *Encrypted Discrete Fourier Transformation (DFT)* (detailed in Section 4.1), *Encrypted Windowing* (detailed in Section 4.2), and the final *Encrypted OCSVM* inference computation (detailed in Section 4.3). The result of the Encrypted OCSVM inference $\widehat{y}$ is still encrypted, and it is finally sent back to the client, where it will be decrypted by using the decryption function $D(\cdot)$ and the secret key, obtaining y.

The EVAD pipeline relies on three key elements.

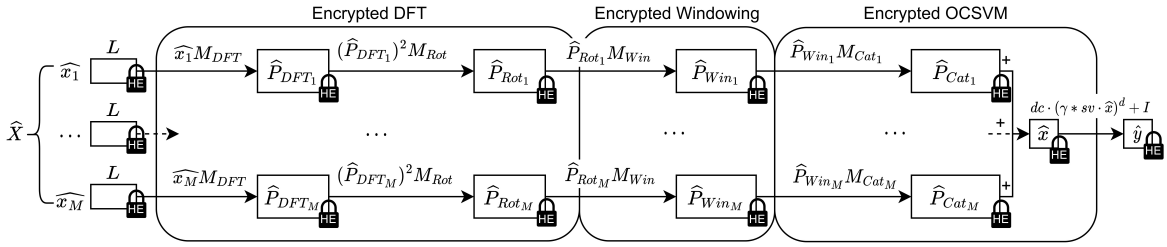

Fig. 2: The encrypted anomaly detection pipeline of the proposed EVAD.

First, the use of the CKKS HE encryption scheme. We opted for this scheme because it can encrypt and natively process real numbers (in particular, it supports the encryption of values in $\mathbb{C}$). Moreover, as anticipated in Section 3, it supports the batching technique, consisting in encoding a vector $\mathbf{z} \in \mathbb{C}^{N/2}$ of size $N/2$ into a polynomial $m(X) \in \mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$. By operating on this polynomial, we can effectively apply the same operation to each of the $N/2$ slots of the ciphertext simultaneously. This greatly reduces both the computational time requested for the processing of encrypted data and the size of the ciphertexts, which is a crucial ability when the encrypted data have to be moved from the client to the third-party. We stress that, as shown in Section 3, the encrypted pipeline needs a set-up phase in which the relinearization and Galois keys are transferred from the client (where the secret and public keys are generated) to the third-party.

Second, the use of OCSVM models. OCSVMs, as anticipated in Section 3.2, are unsupervised ML algorithms for binary classification, commonly used in anomaly detection applications. This is mainly due to their main advantage with respect to supervised learning algorithms, since only data belonging to one class are required for their training. Then, the trained model will be able to distinguish between the data belonging to that class and the data belonging to any other class. They have been chosen in this work because the distance function used for their inference requires only linear operations, hence making it particularly suited to be used in the HE processing of the EVAD pipeline. In this work, the OCSVMs are first trained on plain data, and then used for inference on encrypted data.

Third, the use of a windowing procedure. As shown in Section 4.1, the proposed solution is based on the analysis of the frequency spectrum of the input signals. This requires the computation of the DFT, which is typically highly dimensional, and therefore cannot be used directly as input in a OCSVM. This constitutes a problem from both the computational and algorithmical point of view. The windowing procedure is an effective solution to this issue because it takes the average of all values within a given window, thus reducing the dimension of the inputs to the OCSVM.

We emphasize that in the first two phases (i.e., Encrypted DFT and Encrypted Windowing), each vector $\widehat{x_m}$, $m = 1 \dots M$, is processed independently. This means that these two steps of the pipeline can be applied to each vector $\widehat{x_m}$ in parallel, given that the computation on $\widehat{x_m}$ does not depend on the results of the computation on $\widehat{x_{m'}}$, $m' \neq m$. The outcomes of the Encrypted Windowing are then aggregated before the Encrypted OCSVM phase.

4.1 Encrypted DFT

The first step consists in the encrypted processing of the DFT of each encrypted vector $\widehat{x_m}$.

Given the symmetry of the DFT transformation, only its right part is computed. Hence, the length of the DFT transformation is $L_F = \frac{L}{2} + 1$ if L is even, or $L_F = \frac{L+1}{2}$ if it is odd, where L is the length of $\widehat{x_m}$. The DFT consists in the computation of L_F values in $\mathbb{C}$, out of which the absolute value is taken. In general, the absolute value of a complex value $z = a + bi$ is:

$$|z| = \sqrt{a^2 + b^2}.$$

Unfortunately, the square root operation is not permitted on CKKS vectors, or in any of the HE schemes. In this work, we decided not to replace the square root operation, but to avoid its computation. We emphasize that, as detailed in Section 5, this did not affect the accuracy of the considered OCSVM. Moreover, the computation of the DFT will be carried out by first computing the real and imaginary values of every value of the DFT, then they will be squared and summed up with matrix multiplications. This is allowed by the CKKS scheme since it supports the multiplication between vectors and matrices, as shown in Section 3.1.

Let $M_{DFT_{Real}}$ be a matrix defined as:

$$M_{DFT_{Real}} = \begin{bmatrix} \cos(0) & \cos(0) & \cdots & \cos(0) \\ \cos(0) & \cos\left(\frac{2\pi}{L_F}\right) & \cdots & \cos\left(\frac{2\pi(L_F-1)}{L_F}\right) \\ \cos(0) & \cos\left(\frac{2\pi}{L_F}2\right) & \cdots & \cos\left(\frac{2\pi(L_F-1)}{L_F}2\right) \\ \vdots & \vdots & & \vdots \\ \cos(0) & \cos\left(\frac{2\pi}{L_F}(L_F-1)\right) & \cdots & \cos\left(\frac{2\pi(L_F-1)}{L_F}(L_F-1)\right) \end{bmatrix}, \quad (5)$$

and $M_{DFT_{Im}}$ be a matrix defined as:

$$M_{DFT_{Im}} = \begin{bmatrix} -\sin(0) & -\sin(0) & \cdots & -\sin(0) \\ -\sin(0) & -\sin\left(\frac{2\pi}{L_F}\right) & \cdots & -\sin\left(\frac{2\pi(L_F-1)}{L_F}\right) \\ -\sin(0) & -\sin\left(\frac{2\pi}{L_F}2\right) & \cdots & -\sin\left(\frac{2\pi(L_F-1)}{L_F}2\right) \\ \vdots & \vdots & & \vdots \\ -\sin(0) & -\sin\left(\frac{2\pi}{L_F}(L_F-1)\right) & \cdots & -\sin\left(\frac{2\pi(L_F-1)}{L_F}(L_F-1)\right) \end{bmatrix}. \quad (6)$$

We can then construct M_{DFT} as:

$$M_{DFT} = \begin{bmatrix} M_{DFT_{Real}} & M_{DFT_{Im}} \end{bmatrix} \quad (7)$$

and multiply $\widehat{x}_m$ and M_{DFT} by using Eq. 4. The result $\widehat{P}_{DFT_m}$ of this multiplication will be an encrypted vector of size $L_F \times 2$ storing the real part of the DFT in the left part, and the imaginary part of the DFT in the right one:

$$\begin{aligned} \widehat{P}_{DFT_m} &= \widehat{x}_m M_{DFT} \\ &= [Re_{0,m}, \ldots, Re_{L_F-1,m}, Im_{0,m}, \ldots, Im_{L_F-1,m}]. \end{aligned} \quad (8)$$

To conclude the DFT computation, the values in $\widehat{P}_{DFT_m}$ are initially squared, and then each real value is summed up with its corresponding imaginary value. We emphasize that the square operation is allowed for CKKS vectors since it can be implemented as a simple vector multiplication with itself by using Eq. 2, while, for the sum, a matrix multiplication is used as follows. Let M_{Rot} be:

$$M_{Rot} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad (9)$$

we can use it to sum the real and imaginary values of $\widehat{P}_{DFT_m}$ as:

$$\begin{aligned} \widehat{P}_{Rot_m} &= (\widehat{P}_{DFT_m})^2 M_{Rot} \\ &= [Re^2_{0,m} + Im^2_{0,m}, \ldots, Re^2_{L_F-1,m} + Im^2_{L_F-1,m}]. \end{aligned} \quad (10)$$

If we set $f_{i,m} = Re^2_{i,m} + Im^2_{i,m}$, we can re-write Eq. 10 as:

$$\widehat{P}_{Rot_m} = [f_{0,m}, \ldots, f_{L_F-1,m}]. \quad (11)$$

4.2 Encrypted Windowing

In order to reduce the dimension of the input of the OCSVM model, the encrypted DFT is partitioned into windows and, for each window, the mean of its values is computed.

The goal of the windowing phase is to transform $\widehat{P}_{Rot_m}$, defined in Eq. 11, into a vector of dimension W, where W is a tunable parameter. Such vector $\widehat{P}_{Win_m}$ comprises the following W elements:

$$\widehat{P}_{Win_m} = \left[\frac{\sum_{i=0}^{k-1} f_{i,m}}{k}, \frac{\sum_{i=k}^{2k-1} f_{i,m}}{k}, \ldots, \frac{\sum_{i=(W-1)k}^{Wk-1} f_{i,m}}{k} \right], \quad (12)$$

where $k = \frac{L_F}{W}$, that are computed with a matrix multiplication. Let M_{Win} be a matrix defined as:

$$M_{Win} = \begin{bmatrix} \frac{1}{k} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{k} & 0 & & 0 \\ 0 & \frac{1}{k} & & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & \frac{1}{k} & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & & \frac{1}{k} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & & \frac{1}{k} \end{bmatrix}. \quad (13)$$

We emphasize that M_{Win} has the form of a diagonal matrix of dimension W, having vectors of k $\frac{1}{k}$ values on the main diagonal. Then, we can compute $\widehat{P}_{Win_m}$ as:

$$\widehat{P}_{Win_m} = \widehat{P}_{Rot_m} M_{Win}. \quad (14)$$

We emphasize that the choice of W is problem dependent. This aspect will be addressed in Section 4.5.

4.3 Encrypted OCSVM prediction

The last step of the pipeline is the encrypted OCSVM processing. In this stage, the OCSVM model is used to predict if the encrypted sample is nominal or anomalous.

The first phase of this step is to concatenate the outputs of the pre-processing step (Eq. 14) $\widehat{P}_{Win_m}$, $m = 1 \dots M$, applied to each axis in the input $\widehat{X}$. Being each $\widehat{P}_{Win_m}$ a CKKS vector, it is not possible to just concatenate them as we usually do with normal vectors. A matrix multiplication followed by a sum of vectors is employed to concatenate all the m CKKS vectors $\widehat{P}_{Win_m}$ into a single CKKS vector $\widehat{x}$, representing the input of the OCSVM model.

Let ZM and OM be two matrices defined as follows:

$$ZM = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}; \quad OM = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \quad (15)$$

We can define m matrices M_{Cat_m}, $m = 1 \dots M$, i.e., one for each axis m, which will transform each $\widehat{P}_{Win_m}$ into a vector $\widehat{P}_{Cat_m}$.

M_{Cat_m} is defined as the horizontal concatenation of M ZM matrices, but having the matrix OM in the m-th position:

$$M_{Cat_m} = \Big[ZM \; ZM \; \cdots \; OM \; \cdots \; ZM \; ZM \Big]. \quad (16)$$

Following this notation, the m pre-processed vectors produced in the previous steps are multiplied with the corresponding M_{Cat_m} matrices as follows:

$$\widehat{P}_{Cat_m} = \widehat{P}_{Rot_m} M_{Cat_m}. \quad (17)$$

Then, the input of the OCSVM is computed by summing the M $\widehat{P}_{Cat_m}$ as follows:

$$\widehat{x} = \sum_{m=1}^{M} \widehat{P}_{Cat_m}. \quad (18)$$

We can now compute the encrypted prediction of the model $\widehat{y}$. Let θ be the set of parameters learnt during the training process of the OCSVM. The encrypted distance function $\widehat{y} = f_\theta(\widehat{x})$ of the OCSVM is defined as:

$$\widehat{y} = f_\theta(\widehat{x}) = dc \cdot (\gamma * sv \cdot \widehat{x})^d + I. \quad (19)$$

In more detail, the encrypted input $\widehat{x}$ is initially multiplied with the support vectors sv, by using the CKKS dot-product $\cdot$, shown in Eq. 3. γ is a parameter used in the OCSVM, which is equal to:

$$\gamma = \frac{1}{D \times \sigma^2} \quad (20)$$

where D is the number of features used by the OCSVM, and σ^2 is the variance of the training dataset established during the training phase (i.e., the set of x_i, $i = 1 \dots T$, where T is the number of samples user during the training process).

If we consider that the number of features used by the OCSVM is simply the number of axes M of the samples multiplied by the number of windows W, we can rewrite Eq. 20 as:

$$\gamma = \frac{1}{M \times W \times \sigma^2}. \quad (21)$$

To save one multiplication, we exploit the associative property of multiplication to first compute $\gamma * sv$, given that they are plain values.

Then, we compute the power d operation, as shown in Eq. 19, where d is the degree of the polynomial kernel that is a hyperparameter of the OCSVM. Lastly, the obtained encrypted vector is dot-multiplied with dc, i.e., the vector of the coefficients of the support vectors. The intercept parameter I, which is a trainable parameter, is then added to the final scalar value. The result $\widehat{y}$ can be decrypted by the client to obtain the prediction of the OCSVM.

It is noteworthy that all the operations used in the computation of the distance function of the OCSVM are compatible with the CKKS scheme and in particular with the batching mechanism. We stress that dc, sv, and I are learnt on plain data, while the distance function during the inference is computed on encrypted inputs.

4.4 Pipeline implementation

The pipeline shown in the previous sections allows a third-party to detect anomalies in the encrypted signals coming from a client in a privacy-preserving way, leveraging HE. Unfortunately, as shown in Section 3, HE introduces an important computational overhead in the computation of operations on encrypted data, which is highly dependent on the depth of the pipeline (i.e., the number of consecutive multiplications). We emphasize that the pipeline could be operated by sequentially applying the aforementioned three steps. Nonetheless, we strongly suggest taking advantage of the associative property of matrix multiplication. This allows the third-party service to reduce the multiplicative depth of the pipeline. In particular, the third-party service computes this function for each input $\widehat{x_m}, m = 1 \dots M$:

$$\widehat{P}_{Cat_m} = (\widehat{x_m} M_{DFT})^2 (M_{Rot} M_{Win} M_{Cat_m}). \tag{22}$$

This allows the pipeline to have a lower multiplicative depth, given that the encrypted vector is used in a matrix multiplication (i.e., $\widehat{x_m} M_{DFT}$), then a square operation, and then another matrix multiplication with $(M_{Rot} M_{Win} M_{Cat_m})$. This pipeline has a multiplicative depth of $l = 3$, while in the case in which the associative property of matrix multiplication is not used, such multiplicative depth would be of $l = 5$.

We emphasize that, for the computation of the distance function of the OCSVM (see Eq. 19), additional $2 + (d - 1)$ multiplications are required. This is important for the choice of the CKKS encryption parameters Θ, as described in Section 3.

4.5 Selection of W

In order to select the value of W (i.e., the number of windows to split the DFT of the time-series into), the K-Means [45] clustering method is used. This is done for two reasons. First, during the training phase, only nominal data is available in a classic anomaly detection task, hence it is impossible to tune the value of W according to the accuracy of the OCSVM model on the anomalous data. On the contrary, a clustering approach only requires nominal data, and it provides an initial estimation of the value of W. Second, the K-Means algorithm can leverage the Elbow method [9], which is a method used to estimate the optimal value of K for the clustering.

We emphasize that the choice of W is problem dependent. Therefore, it is impossible to propose a general value of W, optimal for all the cases. The Elbow method for K-Means clustering allows us to set W for a specific dataset. Other solutions could have considered the use of synthetic anomalies in the training set, or, in case some real anomalous data are available, the use of such labeled anomalies for a classification problem.

5 Experimental results

In this section, the proposed EVAD solution is evaluated to assess its effectiveness in a privacy-preserving anomaly detection as-a-service scenario. The code is implemented in Python 3.8. For the OCSVM training, the library scikit-learn [43] is used, while for the CKKS HE operations the TenSEAL [8] library has been used. The experiments have been carried out on an Ubuntu 20.04 LTS workstation equipped with 2 Intel Xeon Gold 5318S CPUs and 384GBs of RAM. First, in Section 5.1 the datasets used in the experimental campaign are detailed. Then, in Section 5.2 the results of the experiments are detailed and commented on. The code to run the experiments is available in the publicly available repository.[2]

[2]The code is attached to this submission, and will be made publicly available in the final version of the paper.

5.1 Considered datasets

5.1.1 Triaxial Bearing dataset

The Triaxial Bearing Vibration dataset [32] is a publicly available[3] dataset collecting triaxial vibration time-series. In particular, the dataset includes time-series collected from a triaxial accelerometer attached to a three phase induction motor. It includes both nominal and anomalous data, artificially induced by introducing faults of increasing severity to the bearings. The faults can be of different depth (expressed in millimeters, or *mm*), with the motor operating under different loads (expressed in Watt, or W), and with different orientations (i.e., inner or outer). A total of 36 classes of anomalies is provided. In our experiments we considered one time-series of nominal data and 36 anomalous time-series, one for each of the 36 anomaly classes. In order to obtain more data points, we split each time-series in subsequences of length 500.

Then, the dataset is divided into training and test sets. The training set refers to the 80% of the nominal data. The test set is composed of two parts: the *nominal test set* consisting in the remaining 20% of nominal data that were not fed to the model for the training and the *anomalous test set* consisting in all the anomalous data that were introduced in the dataset.

5.1.2 CWRU Bearing dataset

The Bearing Data [36] from the Case Western Reserve University (CWRU) is a dataset for bearing fault classification widely used in the literature of vibrational data analysis. It consists of time-series acceleration data collected through uniaxial sensors mounted near to and far from the motor bearings.

For the nominal data, four time-series are provided, corresponding to the four levels of motor loads used in the data collection, i.e., 0HP, 1HP, 2HP, and 3HP. The anomalous data has been collected by inducing faults in three different positions: at the inner raceway, at the rolling element (i.e., the ball), and at the outer raceway. Three different fault diameters were used, i.e., 0.007 inches, 0.014 inches, and 0.021 inches, for the four levels of motor loads. This produced 36 classes of faults. It should be noted that data for the fault of 0.021 inches in the outer position, with 0HP of load, are not available. Both for nominal and anomalous data we split the time-series in windows of 500 points.

[3] http://dx.doi.org/10.17632/fm6xzxnf36.2

Table 1: Number of time-series available for the Triaxial Bearing and CWRU datasets, for the nominal data.

Dataset	Training set	Nominal test set
Triaxial Bearing	189	48
CWRU	2716	679

For the training of the OCSVM, the dataset is divided into training and test sets. The training set refers to the 80% of the nominal data. The test set is composed of two parts: the *nominal test set* consisting in the remaining 20% of nominal data that were not fed to the model for the training and the *anomalous test set* consisting in all the anomalous data that were introduced in the dataset.

The details of the nominal datasets used in the experiments are shown in Tab. 1, while the details of the anomalous datasets are shown in Tab. 2 and Tab. 3 for the Triaxial and CWRU datasets, respectively.

Table 2: Number of time-series available in the anomalous test set for each anomaly class in the Triaxial Bearing dataset.

	Side	Inner			Outer		
	Power	100W	200W	300W	100W	200W	300W
Depth	0.7mm	286	250	227	260	260	132
	0.9mm	304	274	264	281	281	245
	1.1mm	272	274	294	271	303	278
	1.3mm	286	266	255	265	278	255
	1.5mm	262	291	267	263	251	270
	1.7mm	276	275	257	272	261	274

5.2 Experiments

The goal of this experimental campaign is to train EVAD on nominal training data and test it on the encrypted nominal and anomalous test sets. The nominal test sets include data that were not used during the training.

5.2.1 Parameters

More in detail, the OCSVM used in this experiment uses a polynomial kernel with degree $d = 2$, and a value of ν of 5% (i.e., the percentage of training data that is considered anomalous during the training). The method for selecting the value of W to use, introduced in Section 4.5, suggested using a value of $W = 8$ for

Table 3: Number of time-series available in the anomalous test set for each anomaly class in the CWRU dataset.

	Side	Inner				Ball				Outer			
	Power	0HP	1HP	2HP	3HP	0HP	1HP	2HP	3HP	0HP	1HP	2HP	3HP
Depth	0.007"	243	242	242	243	242	241	243	242	241	242	243	243
	0.014"	242	242	242	242	244	243	244	242	243	242	241	241
	0.021"	242	242	241	242	242	242	243	241	/	241	241	242

the Triaxial Bearing dataset and of $W = 10$ for the CWRU dataset.

For what concerns the encrypted processing, the CKKS parameters Θ were chosen to meet the security level of 128−bits and, meanwhile, guarantee a multiplicative depth of 6. Θ was set as follows:

$$\Theta = \begin{bmatrix} N = 16384 \\ b = [60, 50, 50, 50, 50, 50, 50, 60] \\ \Delta = 2^{50} \end{bmatrix}.$$

We stress that, from this settings, the SEAL library was used to compute the actual values for the coefficient modulus list q, starting from N and b.

5.2.2 Accuracy

The model has been initially applied on the nominal data. For what concerns the Triaxial Bearing dataset, it obtained an accuracy of 94.7% on the training set, and of 93.8% on the nominal test set. For the CWRU dataset, it obtained an accuracy of 95.0% on the training set, and of 95.0% on the nominal test set. The values, for both the datasets, are in line with the value of ν used during the training. We stress that only nominal data were used in the training. Then, the same models have been applied on the anomalous test sets. The results, in terms of accuracy (i.e., percentage of anomalies found), are given in Tab. 4 and Tab. 5 for the Triaxial Bearing and CWRU datasets, respectively. Two main comments arise.

First, the EVAD solution is able to recognize the anomalies in the encrypted anomalous test set. In particular, for the Triaxial Bearing dataset, when the anomaly is of low severity (i.e., cases with depth of $0.7mm$), the accuracies are very low for the inner side (ranging from 0.0% to 0.34%). However, when the anomalies become more severe (i.e., cases with depth higher than $0.7mm$), the accuracy quickly reaches 100% highlighting that the OCSVM is able to identify the anomalies in data. Similar considerations hold for the CWRU dataset. In particular, for the Ball type of fault, the anomalies are identified with a high accuracy (100% for the majority of the cases). The OCSVM shows more difficulties in identifying anomalies of low severity (e.g., for the diameter of 0.007 inches of the inner case).

Table 4: Accuracy (in %) for the detection of anomalies by the OCSVM for each anomaly class in the Triaxial Bearing dataset. The percentages are reported both applying the model on plain data (left) and encrypted data (right).

	Power	100W	200W	300W
Depth (mm)	Side	Inner		
	0.7	0.35% / 0.35%	0.80% / 0.80%	0.0% / 0.0%
	0.9	100% / 100%	100% / 100%	100% / 100%
	1.1	100% / 100%	100% / 100%	100% / 100%
	1.3	100% / 100%	100% / 100%	100% / 100%
	1.5	100% / 100%	100% / 100%	100% / 100%
	1.7	100% / 100%	100% / 100%	100% / 100%
	Side	Outer		
	0.7	94.23% / 94.23%	51.15% / 51.15%	63.64% / 63.64%
	0.9	100% / 100%	100% / 100%	100% / 100%
	1.1	100% / 100%	99.34% / 99.34%	100% / 100%
	1.3	100% / 100%	100% / 100%	96.47% / 96.47%
	1.5	99.62% / 99.62%	100% / 100%	100% / 100%
	1.7	100% / 100%	100% / 100%	100% / 100%

Table 5: Accuracy (in %) for the detection of anomalies by the OCSVM for each anomaly class in the CWRU dataset. The percentages are reported both applying the model on plain data (left) and encrypted data (right).

	Power	0HP	1HP	2HP	3 HP
Depth (in)	Side	Inner			
	0.007	0.8% / 0.8%	2.9% / 2.9%	0% / 0%	0% / 0%
	0.014	100% / 100%	100% / 100%	100% / 100%	100% / 100%
	0.021	76.9% / 76.9%	94.2% / 94.2%	88.8% / 88.8%	52.1% / 52.1%
	Size	Ball			
	0.007	100% / 100%	96.3% / 96.3%	100% / 100%	99.6% / 99.6%
	0.014	98.0% / 98.0%	100% / 100%	100% / 100%	99.6% / 99.6%
	0.021	100% / 100%	100% / 100%	100% / 100%	100% / 100%
	Size	Outer			
	0.007	100% / 100%	100% / 100%	100% / 100%	100% / 100%
	0.014	100% / 100%	100% / 100%	73.9% / 73.9%	3.3% / 3.3%
	0.021	NA / NA	100% / 100%	100% / 100%	100% / 100%

Table 6: Computational time (mean and variance) and RAM requirements for the encrypted inference.

Dataset	Computational time (s)	RAM (GB)
Triaxial bearing	7.94 ± 0.12	1.48
CWRU	12.27 ± 0.16	1.56

Second, the accuracy on the plain test set is exactly the same as the one obtained on encrypted data. Tab. 4 and Tab. 5, for each anomaly class, report the plain accuracy (left) and encrypted accuracy (right). The plain accuracy is the accuracy obtained by applying EVAD on the plain inputs, in a classic anomaly detection as-a-service modality. Instead, the encrypted accuracy is the accuracy obtained by applying EVAD on the encrypted inputs. The fact that the plain and encrypted accuracies are always the same, for all the considered classes, highlights that the CKKS scheme is a good choice for EVAD. In detail, the accuracy provided by the CKKS scheme is sufficiently high to guarantee that the introduced rounding errors do not change the outcome of the anomaly detection mechanism when applied on encrypted data. This holds for both the considered datasets.

5.2.3 Efficiency

Tab. 6 details the computational time and memory (RAM) requested for the encrypted processing. We can see that the processing of a single encrypted sample with EVAD, on the machine used for the experiments, requires about 8 seconds for the Triaxial Bearing dataset and of 12 seconds for the CWRU dataset. Moreover, in Tab. 7, the size of all the entities that should be transferred from the client to the third-party and back are shown, with the transfer time computed using a data transfer rate of $1Gb/s$. It should be noted that the auxiliary keys (i.e., relinearization and Galois keys) have to be transferred only once in the set-up phase of the encrypted pipeline. As long as the client does not change the couple of secret and public keys, there is no need to re-transfer them. In the considered scenario, for the Triaxial Bearing dataset, an encrypted sample weighs 17MBs, while for the CWRU this dimension lowers to 11 MBs. The encrypted result weighs 0.9MBs. This is exactly the memory overhead introduced by the CKKS scheme, because the ciphertexts' structure is much richer with respect to the one of the plain values. The transfer times for these entities may be reasonable in a scenario where strict data privacy is required. Lastly, on the third-party service, about $1.5GB$ of RAM are used to process an encrypted sample. This highlights that, as expected, HE introduces a significant overhead in terms of computational demand.

Table 7: Size and transfer time of all the encrypted entities involved in the processing.

Item	Size (MB)	Transfer time (s)
Relinearization and Galois key	349MB	2.9s
Enc. sample (Triaxial Bearing)	17MB	0.14s
Enc. sample (CWRU)	11MB	0.10s
Encrypted answer	0.9MB	0.01s

6 Conclusions

The EVAD solution for the detection of anomalies on vibration encrypted signals proposed in this paper provides strong privacy-preserving capabilities, while maintaining a high level of accuracy. Nonetheless, some limitations are present in our study.

First, some classes of anomalies (often the ones characterizing the weakest anomalies) are often not recognized correctly. While in the majority of classes the accuracy is of 100%, the EVAD's capability in recognizing the weaker anomalies is limited. Other works (e.g., [30] for the CWRU dataset) that are not privacy-preserving can recognize all the anomaly classes with a 100% of accuracy.

Second, the data to train the OCSVM model on has to be provided to the third party in clear. This is necessary given that, for now, EVAD does not allow the learning of a OCSVM directly on encrypted data. This is a well known limitation in the field of privacy-preserving DL with HE.

The next steps in the development of EVAD will be geared toward addressing these two limitations. In particular, to improve EVAD's accuracy, other ML and DL models for the task of anomaly detection will be re-designed to satisfy the constraints imposed by HE. This will enrich EVAD's models repository, potentially making it possible to select the more suited model on the basis of the considered vibrational spectra. Then, the last findings in training encrypted models on encrypted data will be applied to EVAD. This capability can be game-changing in a scenario where clients are not willing to share even nominal data to the third party. Lastly, the application to a different

application scenario will be considered. For instance, the solution proposed in this study can be applied also to the field of Acoustic Anomaly Detection (AAD) due to the similarities between audio and vibration spectra [44].

Data availability

The data used in this study are publicly available at http://dx.doi.org/10.17632/fm6xzxnf36.2 and at https://engineering.case.edu/bearingdatacenter.

Statements

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Acar A, Aksu H, Uluagac AS, et al (2018) A survey on homomorphic encryption schemes: Theory and implementation. ACM Computing Surveys (Csur) 51(4):1–35

[2] Ahmed M, Mahmood AN, Islam MR (2016) A survey of anomaly detection techniques in financial domain. Future Generation Computer Systems 55:278–288

[3] Al Badawi A, Chen L, Vig S (2022) Fast homomorphic svm inference on encrypted data. Neural Computing and Applications 34(18):15555–15573

[4] Alabdulatif A, Kumarage H, Khalil I, et al (2017) Privacy-preserving anomaly detection in cloud with lightweight homomorphic encryption. Journal of Computer and System Sciences 90:28–45

[5] Alabdulatif A, Khalil I, Kumarage H, et al (2019) Privacy-preserving anomaly detection in the cloud for quality assured decision-making in smart cities. Journal of Parallel and Distributed Computing 127:209–223

[6] Alazab M, Gadekallu TR, Su C (2022) Guest editorial: Security and privacy issues in industry 4.0 applications. IEEE Transactions on Industrial Informatics 18(9):6326–6329

[7] Alexandru AB, Burbano L, Çeliktuğ MF, et al (2022) Private anomaly detection in linear controllers: Garbled circuits vs. homomorphic encryption. In: 2022 IEEE 61st Conference on Decision and Control (CDC), IEEE, pp 7746–7753

[8] Benaissa A, Retiat B, Cebere B, et al (2021) Tenseal: A library for encrypted tensor operations using homomorphic encryption. 2104.03152

[9] Bholowalia P, Kumar A (2014) Ebk-means: A clustering technique based on elbow method and k-means in wsn. International Journal of Computer Applications 105(9)

[10] Boemer F, Costache A, Cammarota R, et al (2019) ngraph-he2: A high-throughput framework for neural network inference on encrypted data. In: Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp 45–56

[11] Boulemtafes A, Derhab A, Challal Y (2020) A review of privacy-preserving techniques for deep learning. Neurocomputing 384:21–45. https://doi.org/10.1016/j.neucom.2019.11.041

[12] Campbell C, Bennett K (2000) A linear programming approach to novelty detection. Advances in neural information processing systems 13

[13] Campbell M (2022) Privacy-preserving computation: Doomed to succeed. Computer 55(8):95–99

[14] Chen T, Bao H, Huang S, et al (2022) The-x: Privacy-preserving transformer inference with homomorphic encryption. arXiv preprint arXiv:220600216

[15] Cheon JH, Kim A, Kim M, et al (2017) Homomorphic encryption for arithmetic of approximate numbers. In: International conference on the theory and application of cryptology and information security, Springer, pp 409–437

[16] Christianini N, Shawe-Taylor J (2000) Support vector machines and other kernel-based learning methods. Cambridge UP

[17] Ellefsen AL, Han P, Cheng X, et al (2020) Online fault detection in autonomous ferries:

Using fault-type independent spectral anomaly detection. IEEE Transactions on instrumentation and measurement 69(10):8216–8225

[18] Falcetta A, Roveri M (2022) Privacy-preserving deep learning with homomorphic encryption: An introduction. IEEE Computational Intelligence Magazine 17(3):14–25

[19] Falcetta A, Roveri M (2022) Privacy-preserving time series prediction with temporal convolutional neural networks. In: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8

[20] Falcetta A, Pavan M, Canali S, et al (2023) To personalize or not to personalize? soft personalization and the ethics of ml for health. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, pp 1–10

[21] Fernando T, Gammulle H, Denman S, et al (2021) Deep learning for medical anomaly detection–a survey. ACM Computing Surveys (CSUR) 54(7):1–37

[22] Flynn MJ (1966) Very high-speed computing systems. Proceedings of the IEEE 54(12):1901–1909

[23] Gilad-Bachrach R, Dowlin N, Laine K, et al (2016) Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: International conference on machine learning, PMLR, pp 201–210

[24] Goldwasser S, Micali S (1984) Probabilistic encryption. Journal of Computer and System Sciences 28(2):270–299. https://doi.org/https://doi.org/10.1016/0022-0000(84)90070-9, URL https://www.sciencedirect.com/science/article/pii/0022000084900709

[25] Gorantala S, Springer R, Gipson B (2023) Unlocking the potential of fully homomorphic encryption. Communications of the ACM 66(5):72–81

[26] He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

[27] Hijmans H, Raab CD (2018) Ethical dimensions of the gdpr. Commentary on the General Data Protection Regulation, Cheltenham: Edward Elgar (2018, Forthcoming)

[28] Huang H, Wang Y, Zong H (2022) Support vector machine classification over encrypted data. Applied Intelligence 52(6):5938–5948

[29] Iliadis L, Pimenidis E (2023) Technologies of the 4th industrial revolution with applications. Neural Computing and Applications pp 1–2

[30] Jiang W, Hong Y, Zhou B, et al (2019) A gan-based anomaly detection approach for imbalanced industrial time series. IEEE Access 7:143608–143619

[31] Kiesel R, Lakatsch M, Mann A, et al (2023) Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing. Journal of Cybersecurity and Privacy 3(1):44–60. https://doi.org/10.3390/jcp3010004

[32] Kumar D, Mehran S, Shaikh MZ, et al (2022) Triaxial bearing vibration dataset of induction motor under varying load conditions. Data in Brief 42:108315

[33] Kuo TT, Jiang X, Tang H, et al (2022) The evolving privacy and security concerns for genomic data analysis and sharing as observed from the idash competition. Journal of the American Medical Informatics Association 29(12):2182–2190

[34] LeCun Y, Boser B, Denker JS, et al (1989) Backpropagation applied to handwritten zip code recognition. Neural computation 1(4):541–551

[35] Lee E, Lee JW, Lee J, et al (2022) Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In: International Conference on Machine Learning, PMLR, pp 12403–12422

[36] Loparo K (2012) Case western reserve university bearing data center. Bearings Vibration Data Sets, Case Western Reserve University pp 22–28

[37] Lyubashevsky V, Peikert C, Regev O (2010) On ideal lattices and learning with errors over rings. In: Annual international conference on the theory and applications of cryptographic techniques, Springer, pp 1–23

[38] Lyubashevsky V, Peikert C, Regev O (2013) A toolkit for ring-lwe cryptography. In: Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32, Springer, pp 35–54

[39] Manheim K, Kaplan L (2019) Artificial intelligence: Risks to privacy and democracy. Yale JL & Tech 21:106

[40] Michau G, Fink O (2021) Unsupervised transfer learning for anomaly detection: Application to complementary operating condition transfer. Knowledge-Based Systems 216:106816

[41] Pang G, Shen C, Cao L, et al (2021) Deep learning for anomaly detection: A review. ACM computing surveys (CSUR) 54(2):1–38

[42] Park S, Byun J, Lee J (2022) Privacy-preserving fair learning of support vector machine with homomorphic encryption. In: Proceedings of the ACM Web Conference 2022, pp 3572–3583

[43] Pedregosa F, Varoquaux G, Gramfort A, et al (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830

[44] Pereira PJ, Coelho G, Ribeiro A, et al (2021) Using deep autoencoders for in-vehicle audio anomaly detection. Procedia Computer Science 192:298–307

[45] Saxena A, Prasad M, Gupta A, et al (2017) A review of clustering techniques and developments. Neurocomputing 267:664–681

[46] Schölkopf B, Platt JC, Shawe-Taylor J, et al (2001) Estimating the support of a high-dimensional distribution. Neural computation 13(7):1443–1471

[47] SEAL (2023) Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, microsoft Research, Redmond, WA.

[48] Seeger PM, Yahouni Z, Alpan G (2022) Literature review on using data mining in production planning and scheduling within the context of cyber physical systems. Journal of Industrial Information Integration 28:100371

[49] Sgaglione L, Coppolino L, D'Antonio S, et al (2019) Privacy preserving intrusion detection via homomorphic encryption. In: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, pp 321–326

[50] Smart NP, Vercauteren F (2014) Fully homomorphic simd operations. Designs, codes and cryptography 71:57–81

[51] Trivedi D, Boudguiga A, Triandopoulos N (2023) Sigml: Supervised log anomaly with fully homomorphic encryption. In: International Symposium on Cyber Security, Cryptology, and Machine Learning, Springer, pp 372–388

[52] Ulybyshev D, Bare C, Bellisario K, et al (2020) Protecting electronic health records in transit and at rest. In: 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS), IEEE, pp 449–452

[53] Yuan J, Cao S, Ren G, et al (2022) Lw-net: an interpretable network with smart lifting wavelet kernel for mechanical feature extraction and fault diagnosis. Neural Computing and Applications 34(18):15661–15672

[54] Zheng P, Cai Z, Zeng H, et al (2022) Keyword spotting in the homomorphic encrypted domain using deep complex-valued cnn. In: Proceedings of the 30th ACM International Conference on Multimedia, pp 1474–1483

[55] Zonta T, Da Costa CA, da Rosa Righi R, et al (2020) Predictive maintenance in the industry 4.0: A systematic literature review. Computers & Industrial Engineering 150:106889