CryptIF: Toward Cloud-Based IoT Anomaly Detection Over Encrypted Feature Streams

Teng Li, Zejian Lin, Yebo Feng, Chong Wang, Zhuo Ma, *Senior Member, IEEE*, Bin Xiao, Jianfeng Ma, *Member, IEEE*, and Yang Liu, *Senior Member, IEEE*

***Abstract*—With the proliferation and extensive use of the Internet of Things (IoT), it is vital to ensure the secure operation of IoT devices. However, due to limited computing power and lightweight system design, IoT devices often remain unprotected and are vulnerable to a myriad of attacks. Directly outsourcing the computationally intensive anomaly detection work to some middleboxes or cloud servers seems to solve this problem, but it can raise severe privacy concerns. To tackle these problems, we propose CryptIF, a scalable, privacy-preserving approach to detecting IoT anomalies from the cloud. By leveraging the Extended isolation forest (EIForest) and ciphertext comparison algorithms, CryptIF inspects features encrypted by fully homomorphic encryption (FHE) to detect various IoT anomalies. Furthermore, CryptIF parallelizes computing tasks by taking advantage of the single instruction, multiple data (SIMD) property of Cheon, Kim, Kim and Song (CKKS) homomorphic encryption to accelerate the detection process, thereby significantly increasing its scalability and operating efficiency. The evaluations demonstrate that CryptIF outperforms the state-of-the-art ciphertext-based anomaly detection approach in both detection accuracy and time efficiency. Additionally, CryptIF achieves comparable detection performance to plaintext-based IForest algorithms.**

Received 30 November 2024; revised 10 August 2025; accepted 2 September 2025. Date of publication 12 September 2025; date of current version 10 November 2025. This research was funded in part by the National Key Research and Development Program of China under Grant 2023YFB2904000, in part by the Natural Science Basic Research Program of Shaanxi under Grant 2025JC-JCQN-073, in part by the National Natural Science Foundation of China under Grant 62272370 and Grant 62536002, in part by Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001, in part by the China 111Project under Grant B16037, in part by Qinchuangyuan Scientist + Engineer Team Program of Shaanxi under Grant 2024QCY-KXJ-149, in part by Songshan Laboratory under Grant 241110210200, in part by Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education of China under Grant KLCS20240405, in part by the Fundamental Research Funds for the Central Universities under Grant QTZX23071, in part by the National Research Foundation, Singapore, in part by the Cyber Security Agency under its National Cybersecurity R&D Programme under Grant NCRP25-P04-TAICeN, in part by Ripple under the University Blockchain Research Initiative (UBRI) [1], in part by the National Research Foundation, Singapore, and in part by DSO National Laboratories under the AI Singapore Programme under Grant AISG2-GC-2023-008. Recommended for acceptance by H. Hu. *(Corresponding author: Bin Xiao.)*

Teng Li, Zejian Lin, Zhuo Ma, and Jianfeng Ma are with the School of Cyber Engineering, Xidian University, Xi'an 710126, China (e-mail: litengxidian@gmail.com; 20009201061@stu.xidian.edu.cn; mazhuo@mail.xidian.edu.cn; jfma@mail.xidian.edu.cn).

Yebo Feng, Chong Wang, and Yang Liu are with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: yebo.feng@ntu.edu.sg; chong.wang@ntu.edu.sg; yangliu@ntu.edu.sg).

Bin Xiao is with the School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China, and also with Jinan Inspur Data Technology Company, Ltd., Jinan 250101, China (e-mail: xiaobin@cqupt.edu.cn).

Digital Object Identifier 10.1109/TKDE.2025.3609407

***Index Terms*—Internet of Things (IoT), anomaly detection, isolation forest, homomorphic encryption.**

I. Introduction

THE rapid development and proliferation of Internet of Things (IoT) devices have spawned the rise of IoT-oriented cyber and system attacks. According to recent reports, the market for IoT is expected to grow 16% to 16 billion active connections in 2023 [2], but 57% of the IoT devices are vulnerable to medium or high-severity attacks [3], making IoT security one of the most concerning problems on today's internet. Over the past few years, we have witnessed plenty of large-scale IoT attacks (e.g., Mirai attack [4], RollJam attack [5], Verkada breach [6], etc.). These attacks not only can disturb the operations of IoT devices or steal private data from individuals, but also can cause physical injury to persons, bringing huge damage to both the real and cyber world.

Timely and comprehensive anomaly detection is the key to mitigating IoT attacks [7]. Unfortunately, due to limited computing power and lightweight system design, IoT devices often remain unprotected and are difficult to deploy effective anti-virus or traffic anomaly analysis approaches. To break the computing power limit and provide decent protection to IoT devices, many researchers have proposed to offload the IoT anomaly detection tasks onto some middleboxes or cloud services [8], [9]. Fig. 1 illustrates such an operation model, where a capable cloud server is equipped with sate-of-the-art anomaly detection approaches and the correlative IoT devices keep uploading their operational data for troubleshooting.

However, as illustrated in Fig. 1, there are huge privacy risks lurking in such models. As all the data will be converged to the cloud server for anomaly detection, the server itself becomes a single point of failure. Once any malicious third party successfully compromises the server, all the private IoT data of users will be leaked. Besides, the cloud server may be managed by some dishonest administrators, who would steal or inspect users' data simultaneously with the data processing for malicious purposes, such as data reselling, phishing, identity tracking, illegal analysis, etc.

To resolve the aforementioned problems, researchers can utilize privacy-preserving computing schemas (i.e., differential privacy (DP) [10], secure multi-party computation (MPC) [11], homomorphic encryption (HE) [12]) to enable a third party to inspect the data without disclosing private information to it. Nonetheless, existing approaches will inevitably incur many

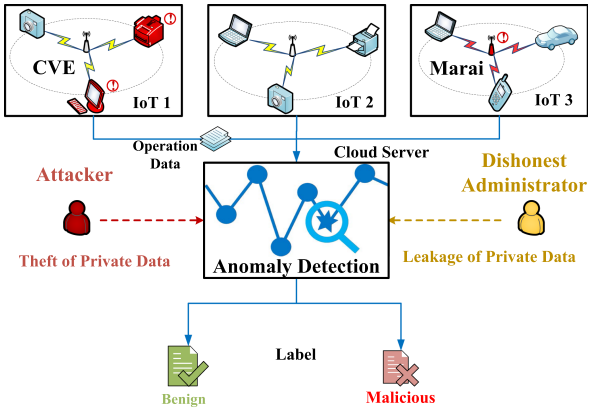


Fig. 1. The operation model of traditional cloud-based IoT anomaly detection approaches.

side effects, making them hardly applicable in real-world IoT scenarios. For example, DP-based approaches insert noise in raw data to prevent information leakage, which brings a noticeable decrease to the detection efficacy and cannot fundamentally solve the privacy issue [10], [13]; MPC-based approaches consume high computing resources on both parties (i.e., IoT devices and the cloud server); approaches based on HE also face efficacy drops and scalability issues if not properly designed.

To fill the gap, we propose CryptIF, a cloud-based IoT anomaly detection approach that can simultaneously preserve user privacy, achieve decent detection efficacy, and maintain high operating efficiency. By leveraging fully homomorphic encryptions (FHE), CryptIF only analyzes encrypted features streamed by IoT devices to detect anomalies. To conduct accurate detection over ciphertexts, we develop an improved detection approach based on the Isolation Forest (IForest) [14], which is an efficient unsupervised learning method for outlier detection. In result, CryptIF can accurately identify various anomalies, including both IoT system anomalies (e.g., sensor faults, firmware hijacking, eavesdropping, etc.) and network anomalies (e.g., DDoS, botnet, port scan, etc.). In addition, we have adopted a variety of measures to improve the operating efficiency of the system, making CryptIF deployable on both the IoT devices and the cloud server.

Compared with existing solutions towards privacy-preserving IoT anomaly detection, our work makes the following contributions:

- By leveraging FHE, we have proposed the first privacy-preserving training and prediction process for the IForest algorithm. CryptIF can radically eliminate the privacy concerns during the whole anomaly detection process. Even if the cloud server is compromised, the users' data still remain protected.
- We have enhanced the training process of the IForest algorithm by merging multiple encrypted inner products through unified transmission, achieving node separation. This design yields significant improvements in both the construction time of the forest and the detection efficiency.
- CryptIF is designed for parallel processing. Whether it is the multi-node inner product parallel calculation during the training process or the multi-tree parallel calculation in the anomaly detection process, the efficiency of anomaly detection can be significantly enhanced while protecting privacy. This feature meets the needs of detecting multiple IoT devices.
- CryptIF adopts multiple measures to increase the operation efficiency for both the IoT devices and the cloud server. It utilizes statistical approaches to select high-quality features from the raw data, which reduces the sizes of messages to be encrypted and transmitted; To accelerate the detection process and increase the scalability, the cloud server parallelizes detection tasks through the Comp algotithm [15], which takes advantage of the single instruction, multiple data (SIMD) property of Cheon, Kim, Kim and Song (CKKS) HE.

We evaluated CryptIF using anomaly detection datasets from public repositories and found that it achieves high efficiency and strong detection efficacy. CryptIF is capable of achieving line-rate performance for both feature encryption and anomaly detection tasks, and consistently outperforms existing ciphertext-based anomaly detection approaches across a variety of anomaly types.

In this work, we primarily focus on protecting user data privacy during the transmission and cloud processing phases of IoT anomaly detection. CryptIF is designed to address the risks of data leakage stemming from potentially compromised or semi-honest cloud servers. By operating entirely over encrypted feature streams, CryptIF ensures that even if the cloud server is breached, sensitive information about IoT device operations remains protected.

II. Related Work

In this section, we enumerate research works related to cloud-based privacy-preserving anomaly detection.

A. Privacy-Preserving Computation Frameworks

The current privacy-preserving computation frameworks can be classified into three categories: DP, MPC, and FHE. DP adds noise to raw data to mask the actual attribute values. Although DP-based approaches can carry out privacy-preserving computations [16], they are unable to fully prevent privacy leaks and will bring noticeable drop to the calculation accuracy due to the added noise. MPC allows two or more parties to collaborate to compute a function without leaking the function's input [17], but it is costly in terms of the required computing power on all the parties engaged.

FHE enables different parties to carry out computations on encrypted data. FHE was first proposed by Gentry in 2009 [12]. Current widely-used FHE algorithms implemented in major open source libraries include CKKS [18], BGV [19], BFV [20], TFHE [21], etc. Among them, TFHE is based on boolean circuit; BGV and BFV are based on arithmetic circuit and they only support encrypting integers; CKKS can handle approximate floating point numbers. We compare the capabilities of these commonly-used FHEs in Table I. The SIMD technique refers to paralleizing computing tasks over ciphertexts, thereby reducing the time consumption [22]. However, the operations of FHE are limited to addition/subtraction and multiplication of ciphertext.

TABLE I
COMPARISON OF EXISTING FHE SCHEMES

Operation	CKKS [17]	BGV [18]	BFV [19]	TFHE [20]
Native Add/Sub	✓	✓	✓	✓
Native Mult	✓	✓	✓	✓
Floating Point	✓	✗	✗	✗
SIMD	✓	✓	✓	✗

Thus, unlike DP and MPC, it is difficult for FHE to carry out ciphertext comparison. In 2019 Cheon et al. [23] first proposed the numerical method for comparison on CKKS ciphertext. Later, Cheon et al. [15], [24] proposed efficient homomorphic comparison methods with optimal complexity, which is 30 times faster than the previous approach.

B. Detection Solutions for Privacy-Preserving Purposes

Based on the aforementioned privacy-preserving computation frameworks, researcher proposed various approaches to detecting anomalies without inspecting the raw data [25], [26], [27], [28], [29], [30], [31], [32], [33], thereby eliminating the privacy concerns.

In 2017, Alabdulatif et al. [34] proposed the first anomaly detection approach in cloud using lightweight HE. However, the Domingo-Ferrer HE, which was used in this approach, has been proved to be difficult to resist known-plaintext attacks [35]. Therefore, in 2019, Alabdulatif et al. [36] proposed another anomaly detection based on BGV, using the FCM clustering to detect anomalies over ciphertexts. To solve the problem that BGV cannot encrypt floating-point numbers, they use the IEEE 754 standard [37] to represent floating-point numbers. In 2019, Du et al. [38] proposed a DP-based anomaly detection, which applies DP noise to help identify outliers and novelty points. However, it requires manually configuring different noise for different datasets to ensure the detection accuracy. In 2020, Li et al. [39] proposed a medical detection approach based on MPC. However, to support operations of multi-party computing, MPC consumes a large amount of computing and communication resources. It is difficult to deploy such an approach on IoT devices due to limited computing power. In 2021, Itokazu proposed a decision-tree-based anomaly detection approach using HE [40]. This approach leverages a federated learning scheme to construct ensemble decision trees for detection. As this approach requires each participated party to built the decision tree, which is computationally intensive, this approach is not suitable for IoT environments. In 2022, Kurt et al. proposed a distributed differentially private generalized CUSUM detector [41]. This detector infers network anomalies from perturbations and encrypted messages received from nodes. However, the differential privacy computation in distributed systems is complex, and particularly when handling large-scale data streams, high computational complexity may lead to decreased system performance and increased latency. In 2023, Arazzi et al. proposed a Federated Learning-based anomaly detection approach using HE [42]. However, in some cases, there is a significant communication overhead involved if the anomaly detection model is large and the synchronization among parties needs to occur frequently.

III. DESIGN OF CRYPTIF

This section details the architecture, workflow, and algorithms of CryptIF, a cloud-based IoT Anomaly detection approach over encrypted feature streams.

Fig. 2 illustrates the architecture and workflow of CryptIF. CryptIF inspects the operating data from IoT devices to conduct the anomaly detection, such data include traffic data from the IoT network and system data from the IoT devices. These data may include but not limited to temperature, humidity, and light hours collected by sensors and the source and destination IP addresses, the source and destination port numbers, the number of connections, the protocol type, the packet sizes in network traffic.

Initially, each protected IoT device is responsible for real-time collection of system operation data and network traffic data as raw features. Due to the typically limited computational and storage capabilities of IoT devices, complex encryption operations are not performed locally. After data collection, IoT devices establish lightweight TLS or DTLS encrypted communication channels to upload the collected raw data to nearby deployed edge nodes. To ensure the confidentiality and integrity of feature streams transmitted from IoT devices to edge nodes, CryptIF utilizes lightweight TLS/DTLS protocols. These protocols are widely used in resource-constrained IoT environments and are standardized in documents such as RFC 6347 (DTLS 1.2) and RFC 8446 (TLS 1.3), which provide secure communication with minimal overhead. The edge nodes, serving as intermediate processing layers, receive and buffer data streams from multiple IoT devices.

In the CryptIF system, the edge node serves as a crucial intermediary between IoT devices and the cloud server, undertaking key roles in data pre-processing and secure computation assistance. Specifically, the edge node performs the following functions: First, it receives feature data streams uploaded from IoT devices. These raw data streams are subjected to initial cleaning operations, including normalization, missing value imputation, and basic outlier removal, to ensure data quality and consistency. Subsequently, the edge node conducts lightweight feature selection, primarily by eliminating low-variance features and prioritizing features with high kurtosis, thereby enhancing the performance of the subsequent anomaly detection model. After feature selection, the edge node encrypts the refined feature vectors using fully homomorphic encryption (FHE) schemes and uploads the ciphertexts to the cloud server, thereby ensuring end-to-end data confidentiality throughout the computation process. In addition to its pre-processing tasks, the edge node also participates in assisting encrypted comparison operations. During the training and evaluation phases of the anomaly detection model, when the cloud server requires node-splitting decisions, the relevant ciphertext pairs are sent back to the edge node. The edge node temporarily decrypts these ciphertexts, performs the necessary comparison operations, and returns only the comparison outcomes (e.g., 0 or 1), without exposing any plaintext feature values. To prevent side-channel attacks and mitigate risks associated with key management during the decryption and comparison processes, the edge node optionally employs a Trusted Execution Environment (TEE), such as Intel

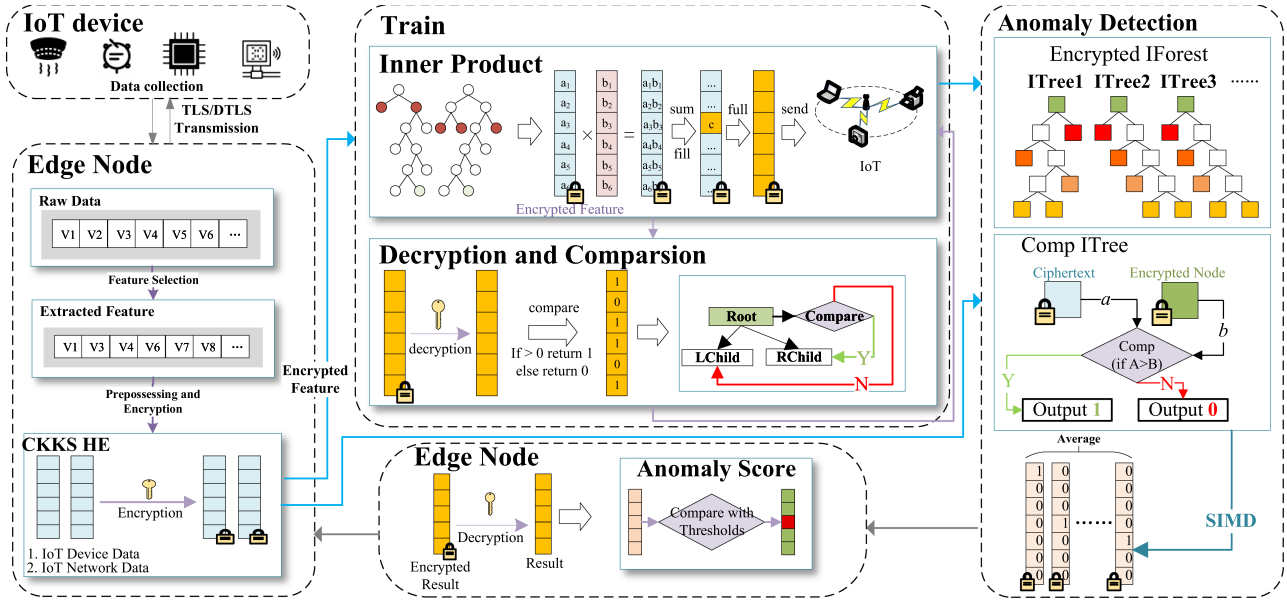


Fig. 2. The architecture and workflow of CryptIF.

SGX or ARM TrustZone. By confining critical operations within the TEE, CryptIF effectively reduces the threat of side-channel attacks and ensures secure key management, thereby further strengthening the system's end-to-end security guarantees.

CryptIF on the server side includes two modes, as shown in Fig. 2: training mode and detection mode. The former uses unlabeled IoT data to derive tree-based anomaly detection models, while the latter uses the detection model to detect IoT anomalies on encrypted feature streams. In the training mode, uploaded encrypted feature streams are used to train the EIForest clustering model. By using the inner product of encrypted vectors and storing multiple inner products in the same vector, which is sent to the edge node for unified decryption, the segmentation nodes can be determined based on the difference between the returned features. Through these operations, CryptIF can significantly reduce construction time and improve detection efficiency. In the detection mode, CryptIF utilizes the trained model to calculate a result vector containing necessary information to help IoT devices determine the presence of anomalies. By utilizing the CKKS HE's ciphertext comparison and SIMD features, the detection pipeline can be parallelized to speed up the entire analysis process. It is worth noting that all of these operations are performed on ciphertext, not plaintext.

After the cloud server derives the result vector (in encrypted form), it sends it back to the corresponding edge node. The edge node then decrypts the encrypted result vector and calculates the anomaly score to determine whether an anomaly is present.

A. CKKS Fully Homomorphic Encryption

The CKKS algorithm, also known as HEAAN algorithm, is a HE scheme based on approximate numerical algorithm, which provides homomorphic approximation algorithm for floating-point type data, capable of encrypting real or complex numbers and satisfying efficient approximation operations, and can be applied to some real-life needs. Thus, compared with the BGV algorithm and the BFV algorithm, the biggest advantage of the CKKS algorithm is that it can handle floating-point numbers, even complex numbers, and its plaintext field is a complex number vector. It is also recommended for encrypted data processing in data science and machine learning.

Most existing schemes rely on noisy encryption, where a small amount of noise is intentionally added during the encryption process to obscure the original message. The noise budget is predetermined during the scheme's initialization, and each homomorphic operation on the ciphertext consumes a portion of this budget. As computations accumulate, the noise level increases, gradually degrading the precision of the encrypted message. In contrast, the CKKS scheme encodes encrypted values as approximate numbers with an initial level of precision. The injected noise remains smaller than this predefined precision, thereby allowing for controlled accuracy throughout computation. The following outlines several details of the CKKS scheme:

The noise budget for the CKKS scheme is initialized with the parameter L. For every multiplication, the noise budget is subtracted by the integer p. The noise budget for a given ciphertext is denoted as l. When the message is just encrypted, $l = L$. When $l < p$, the noise budget is said to be deplete.

1) KeyGen: Let 2^L be the initial ciphertext mode, and let $HWT(h)$ denote the distribution of a polynomial uniformly chosen from R_{2^L} with coefficients h and h is non-zero. a secret s is drawn from $HWT(h)$, a random number a from $U(R_{2^L})$, and an error e from $DG(\sigma^2)$. Set the key to $sk(sk \leftarrow (1, s))$ and the public key to $pk(pk \leftarrow (b, a) \subseteq R_{2^L})$, where $b = -a \cdot s + e(\mathrm{mod}L)$. Then take $a' \leftarrow U(R_{2^L})$, $e' \leftarrow DG(\sigma^2)$, and set the evaluation key to $evk(evk \leftarrow (b', a'))$, where $b' = -a' + e' + L \cdot s^2(\mathrm{mod}2^L)$.

2) Encrypt(pk, m): For $m \subseteq R_{2^L}$, sample $v \leftarrow U(R_{2^L})$, $e_0, e_1 \leftarrow DG(\sigma^2)$, and the result after encryption is $ct = -v \cdot pk + (m + e_0, e_1)(\mathrm{mod}2^L)$.

3) Decrypt(sk, ct): For $ct = ((c_0, c_1), l) \subseteq R_{2^l}$, the result after decryption is $c_0 + c_1 \cdot sk (\mathrm{mod} 2^l)$.

4) Add(ct_1, ct_2): For the ciphertext $ct_1 = ((c_0, c_1), l) \subseteq R_{2^l}$, $ct_2 = ((c'_0, c'_1), l) \subseteq R_{2^l}$, the ciphertext $ct = (c_0, c_1) + (c'_0, c'_1) (\mathrm{mod} 2^l)$ is output.

5) Mult(ct_1, ct_2): For the ciphertext $ct_1 = ((c_0, c_1), l) \subseteq R_{2^l}$, $ct_2 = ((c'_0, c'_1), l) \subseteq R_{2^l}$, then let $(d_0, d_1, d_2) = (c_0 c'_0, c_1 c'_0 + c_0 c'_1, c_1 c'_1) (\mathrm{mod} 2^l)$, the ciphertext $ct = (d_0, d_1) + [2^{-L} \cdot d_2 \cdot evk (\mathrm{mod} 2^l)]$ is output.

6) Rescale(ct, p): For $ct = ((c_0, c_1), l) \subseteq R_{2^l}$, an integer $p \leq l$, the ciphertext is rescaled and output $ct' = [2^{-p} \cdot (c_0, c_1)] (\mathrm{mod} 2^{l-p})$

7) Rotate(ct, r): For the ciphertext vector ct, the output ct vector shifts r positions to the right of the vector ct'.

The above is the CKKS algorithm under asymmetric HE, but in practice CKKS encryption can be either asymmetric or symmetric. Since the evaluation key means that the computation of the encrypted data (usually executed by an untrusted party) is performed to produce the encrypted result. Therefore, it is only necessary that the decryption algorithm is performed by the trusted party that has access to the key. For the symmetric encryption algorithm of CKKS, the encryption process and decryption process are as follows:

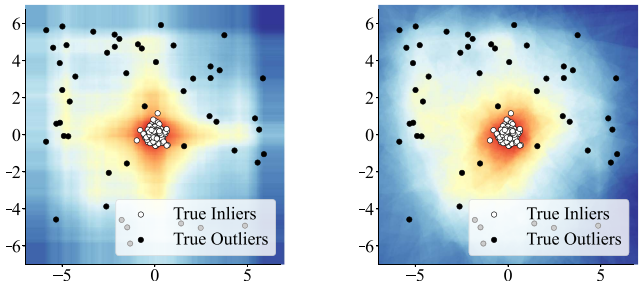
8) SymEncrypt(m, sk): For $m \subseteq R_{2^L}$, sample $a \leftarrow U(R_{2^L})$, $e \leftarrow DG(\sigma^2)$, and the result after encryption is $ct = -a \cdot sk + m + e (\mathrm{mod} 2^L)$.

9) SymDecrypt(ct, sk): For $ct = (c, l) \subseteq R_{2^l}$, the result after decryption is $c \cdot sk (\mathrm{mod} 2^l)$.

In addition, CKKS supports SIMD technology, which can significantly reduce computational overhead when the cloud server performs homomorphic computations on ciphertext vectors. The CKKS scheme utilizes a polynomial ring structure to encode multiple scalar values into a single ciphertext. This encoding method allows a single operation to be executed on multiple data points simultaneously, replacing multiple identical homomorphic operations with a single one. For example, to perform homomorphic multiplication on two sets of ciphertexts $[\![\mathbf{a}]\!] = ([\![a_1]\!], [\![a_2]\!], \ldots, [\![a_n]\!])$ and $[\![\mathbf{b}]\!] = ([\![b_1]\!], [\![b_2]\!], \ldots, [\![b_n]\!])$, without using SIMD, the computation would be $[\![\mathbf{a} \cdot \mathbf{b}]\!] = ([\![a_1 \cdot b_1]\!], [\![a_2 \cdot b_2]\!], \ldots, [\![a_n \cdot b_n]\!])$. This requires executing n multiplication operations. However, if SIMD is used, the plaintexts $\mathbf{a}$ and $\mathbf{b}$ can be encoded as a and b respectively, and only a single homomorphic multiplication operation is needed to obtain the same result $[\![a \cdot b]\!] \leftrightarrow [\![\mathbf{a} \cdot \mathbf{b}]\!] = ([\![a_1 \cdot b_1]\!], [\![a_2 \cdot b_2]\!], \ldots, [\![a_n \cdot b_n]\!])$.

B. Extended Isolation Forest

IForest isolates and identifies anomalies by randomly partitioning the data space, leveraging the randomness of feature and value selection. Due to the rarity and uniqueness of anomalies, they quickly stand out in these random selections. The traditional Isolation Forest algorithm randomly selects a feature for each split, producing axis-parallel hyperplanes. These hyperplanes may overlap, making it difficult to detect local outliers. A heatmap can display anomaly scores in different regions. The closer the color of a region is to blue, the more likely the points in that region are anomalies. Conversely, the closer the color is to red, the more likely the points in that region are normal data points. As shown in Fig. 3(a), the shape of the delineated normal data region is a 'cross-star,' and some local outliers also have relatively high anomaly scores. The EIForest [43] replaces the axis-parallel hyperplanes with hyperplanes of random slopes, as shown in Fig. 3(b). This method introduces more randomness in each hyperplane split, better delineating the regions in the normal dataset, effectively addressing local anomalies and overcoming the limitations of the traditional IForest algorithm.



(a) IForest. (b) EIForest.

Fig. 3. In the heatmap, areas with redder colors indicate higher anomaly scores. Compared to IForest, EIForest is better at outlining the regions where normal data is concentrated.

C. Ciphertext Comparison

In the IForest, it is essential to carry out value comparisons both during the training and detection modes. However, it is not easily achievable over ciphertexts. Thus, it is important to design and implement an effective ciphertext comparison approach in CryptIF. In general, the comparison of two numbers a and b is based on (1a), (1b), or (1c).

$$\mathbf{Max}(a, b) = \frac{(a+b)}{2} + \frac{|a-b|}{2}, \tag{1a}$$

$$\mathbf{Min}(a, b) = \frac{(a+b)}{2} - \frac{|a-b|}{2}, \tag{1b}$$

$$\mathbf{Comp}(a, b) = \frac{\mathrm{sgn}(a-b)+1}{2} = \begin{cases} 1 & \text{if } a > b \\ 1/2 & \text{if } a = b \\ 0 & \text{if } a < b \end{cases}. \tag{1c}$$

The **Max/Min** can return the larger or smaller values of the two numbers. The **Comp** is able to determine the relationship between the sizes of two numbers. Cheon et al. [15] proposed an efficient homomorphic comparison method with optimal complexity, which is more efficient than the previous method. Therefore, we use this ciphertext comparison method to implement the comparison operation in CryptIF. The ciphertext comparison is used in both the training and detection modes of CryptIF. We introduce the detailed ciphertext comparison procedure as follows.

For (1), it is difficult to fit the $\mathrm{sgn}(x)$. By polynomial fitting method, we can generate a $f(x)$ to fit $\mathrm{sgn}(x)$. Thus, the $f(x)$ can be calculated as in (2).

$$f_n(x) = \begin{cases} \frac{x^2}{(x^2+(1-x)^2)} & \text{if } n = 0, \\ \sum_{i=0}^{n} \frac{1}{4^i} \cdot \binom{2i}{i} \cdot x \left(1 - x^2\right)^i & \text{if } n > 0, \end{cases} \tag{2}$$

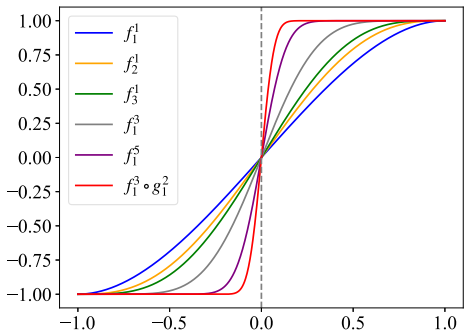


Fig. 4. With the increasing of n, $f(x)$ will gradually fit $\mathrm{sgn}(x)$. Through composition of functions, with the increasing of d, $f(x)$ will also gradually fit $\mathrm{sgn}(x)$. Besides, u is better than n in fitting $\mathrm{sgn}(x)$.$f^{(u_f)} \circ g^{(u_g)}(a-b)$ is better than ${f_n}^{(u)}$ in fitting $\mathrm{sgn}(x)$ with the same n and u values.

where the highest index number of $f_n(x)$ is $2n+1$. Therefore, the larger n is, the greater the ability it has to fit the $\mathrm{sgn}(x)$. However, the larger value of n, the greater computational complexity it brings. Thus, multiple $f(x)$ can be compounded, e.g., ${f_n}^{(u)}(x) = f_n \circ f_n \circ f_n \cdots \circ f_n(x)$, where the u denotes the number of composition times. As illustrated in Fig. 4, with the increasing of n, $f(x)$ will gradually fit $\mathrm{sgn}(x)$ [15]. Furthermore, through composition of functions, with the increasing u, $f(x)$ will also gradually fit $\mathrm{sgn}(x)$. Besides, we can see that u works better than n in fitting $\mathrm{sgn}(x)$. Therefore, choosing the suitable n and u enables it to fit $\mathrm{sgn}(x)$ by ${f_n}^{(u)}(x)$. The **Max/Min** and **Comp** can be calculated using $f(x)$ as (3a), (3b), and (3c).

$$\mathbf{Max}(a,b) = \frac{a+b}{2} + \frac{a-b}{2} \cdot f_n^{(u)}(a-b), \tag{3a}$$

$$\mathbf{Min}(a,b) = \frac{a+b}{2} - \frac{a-b}{2} \cdot f_n^{(u)}(a-b), \tag{3b}$$

$$\mathbf{Comp}(a,b) = \frac{\mathrm{sgn}(a-b)+1}{2} \approx \frac{f_n^{(u)}(a-b)+1}{2}. \tag{3c}$$

In order to improve the fitting accuracy and reduce the computational complexity, Cheon et al. proposed the $g(x)$, which can be compounded with $f(x)$ to improve the fitting accuracy of the fitting $\mathrm{sgn}(x)$. Like $f(x)$, the $g(x)$ can also be optimized by n and d. As illustrated in Fig. 4, with the same n and number of composition times u, $f_1^{(3)} \circ g_1^{(2)}$ performs better than $f_1^{(5)}$ in fitting $\mathrm{sgn}(x)$. For ease of computation, the common $g(x)$ functions are listed follows.

$$g_1(x) = -\frac{1359}{2^{10}} \cdot x^3 + \frac{2126}{2^{10}} \cdot x,$$

$$g_2(x) = \frac{3796}{2^{10}} \cdot x^5 - \frac{6108}{2^{10}} \cdot x^3 + \frac{3334}{2^{10}} \cdot x,$$

$$g_3(x) = -\frac{12860}{2^{10}} \cdot x^7 + \frac{25614}{2^{10}} \cdot x^5 - \frac{16577}{2^{10}} \cdot x^3 + \frac{4589}{2^{10}} \cdot x.$$

Therefore, we can choose the suitable n and u to fit $\mathrm{sgn}(x)$ by $f_{n_f}^{(u_f)} \circ g_{n_g}^{(u_g)}$. The **Max/Min** and **Comp** can be calculated using $f \circ g(x)$ as (4a), (4b), and (4c).

$$\mathbf{Max}(a,b) = \frac{a+b}{2} + \frac{a-b}{2} \cdot f^{(u_f)} \circ g^{(u_g)}(a-b), \tag{4a}$$

Algorithm 1: ArrayMax.

Input: $a_1, a_2, \ldots, a_m \in [0,1)$,comparison parameter $n, u \in N$.
Output: an approximate value of $max(a_1, a_2, \ldots, a_m)$.

```
//Create a binary array.
a_{1,0}, a_{2,0}, ..., a_{m,0} ← (a_1, a_2, ..., a_m),
k ← m,
for j ← 0 to ⌊log_2 m⌋ do
    if k is odd then
        a_{[k/2],j+1} ← a_{k,j}.
    end if
    k ← ⌊k/2⌋.
    for i ← 1 to k do
        //Use (4a).
        a_{i,j+1} ← Max (a_{2i-1,j}, a_{2i,j}; n, u_f, u_g).
    end for
end for
return a_{1,⌊log_2 m⌋}.
```

$$\mathbf{Min}(a,b) = \frac{a+b}{2} - \frac{a-b}{2} \cdot f^{(u_f)} \circ g^{(u_g)}(a-b), \tag{4b}$$

$$\mathbf{Comp}(a,b) = \frac{\mathrm{sgn}(a-b)+1}{2} \approx \frac{f^{(u_f)} \circ g^{(u_g)}(a-b)+1}{2}. \tag{4c}$$

Therefore, there are two ways to implement ciphertext comparison in CKKS. In CryptIF, we need to obtain the maximum or minimum value in encrypted streams when isolation tree (ITree) separates the left or right node. Thus, by using Algorithm 1, CryptIF can obtain the maximum value in the ciphertext array. The **ArrayMin** algorithm is similar to the **ArrayMax**.

D. Extended IForest Train on HE

In CryptIF, the cloud server utilizes encrypted data uploaded by IoT devices to train an Extended IForest model, thereby generating an encrypted anomaly detection model. However, we identify two sources of randomness inherent in the traditional Isolation Forest algorithm that may compromise its effectiveness and lead to detection errors. The first source of randomness occurs during the training phase, where features are randomly selected for node splitting. Since not all features possess strong discriminative power, arbitrary feature selection may hinder the model's ability to effectively separate normal and anomalous instances. In other words, the original dataset may contain redundant or irrelevant features that are not informative for anomaly detection and may even introduce noise, adversely affecting the model's performance. To mitigate the adverse effects caused by this randomness, we introduce statistical techniques during the model initialization phase to identify and retain high-quality features from the original data. Specifically, we adopt a two-step feature selection strategy combining variance thresholding and kurtosis-based selection. First, features with low variance—typically below a predefined threshold (e.g., 0.01)—are removed, as they exhibit little variability across samples and contribute minimal information. Next, we compute

the kurtosis of the remaining features and rank them according to their kurtosis values. The top k features with the highest kurtosis are selected for model training. It is worth noting that kurtosis measures the "peakedness" of a feature's distribution. In the context of anomaly detection, high kurtosis often indicates the presence of extreme values or abrupt deviations, which are characteristic signatures of anomalies. Therefore, features with higher kurtosis are more likely to highlight abnormal patterns in the data, thereby enhancing the model's sensitivity and detection accuracy. Moreover, selecting informative features reduces the dimensionality of the encrypted data, leading to lower computational costs and reduced communication overhead in the privacy-preserving setting. In summary, the proposed feature selection strategy not only achieves dimensionality reduction but also enhances information density, thereby improving the performance, communication efficiency, and computational feasibility of the Extended Isolation Forest model under encryption. This provides a more robust foundation for subsequent anomaly detection tasks.

When training an Extended IForest with encrypted data, it is necessary to compute the inner products of the split data and compare them with the split tree nodes. Since the cloud server cannot directly determine the relative sizes of encrypted values, communication with the corresponding edge node is required during this process. Specifically, the cloud sends the encrypted comparison data to the designated edge node, which decrypts the inner product results and returns the comparison outcomes to the cloud for subsequent tree construction. To reduce communication overhead and avoid wasting cloud resources, all trees perform comparison calculations collectively at each training layer. Thus, the inner products of all nodes requiring comparison across all isolation trees are batched together, transmitted to the edge node for decryption, and the decrypted comparison results are sent back to the cloud server.

During the training process of an extended isolation forest based on encrypted data, it is necessary to compute the inner product of the data and compare it with the split nodes of the decision trees. Since the data is encrypted, the cloud server cannot directly read the data values and must rely on interaction with the edge node to complete the comparison operations. However, this interaction introduces two major challenges: (i) frequent data exchanges result in significant communication overhead; (ii) if each tree in the isolation forest is computed separately, computing resources may be idle while waiting for data to be returned, reducing overall efficiency. To reduce communication overhead and avoid wasting cloud resources, we propose a communication packaging process. This method leverages the parallel tree structure of the Isolation Forest model to minimize resource consumption and communication cost through encrypted vector encapsulation. When all trees are trained to a certain depth, they simultaneously require comparison operations for all nodes at that level. Therefore, the inner products of all nodes requiring comparison across all isolation trees can be batched together and sent to the corresponding edge node for decryption and comparison result return. First, the encrypted data is processed using the ciphertext inner product algorithm for CKKS vectors proposed by Benaissa et al. [44]. This algorithm exploits the rotatability property of CKKS to shift and rotate ciphertexts, iteratively accumulating the inner product results such that each position of the ciphertext vector holds the same inner product value. For the ith tree, the inner product result is multiplied by a unit vector with a one at the ith position and zeros elsewhere. In this way, the ith position of the resulting ciphertext vector contains the inner product for the ith tree, while all other positions remain zero. By distributing the inner product results across different positions, they can be combined into a single ciphertext vector. This ciphertext vector is then transmitted to the edge node, which performs decryption and returns the comparison results to the cloud server. Algorithm 2 specifically describes the ciphertext inner product and packaging process. This solution enables unified decryption and result aggregation, thereby improving the overall efficiency of cloud-side processing.

Algorithm 2: Encrypted Inner Product Result Packaging.

Input: The ciphertext vector $\vec{a_i}$, ciphertext vector $\vec{b_i}$ of the ith node in the Extended Isolation Forest training process,vector length N,the number of trees K.
Output: Ciphertext vector inner product $[\vec{c}]$.

for $i \leftarrow 1$ to K **do**
  $\vec{a_i} = [a_{i1}, a_{i2}, \ldots, a_{iN}]$,
  $\vec{b_i} = [b_{i1}, b_{i2}, \ldots, b_{iN}]$,
  $\vec{x} = [0, \ldots, i, \ldots, 0]$,
  $\vec{c_i} = \vec{a_i} \times \vec{b_i} = [a_{i1}b_{i1}, a_{i2}b_{i2}, \ldots, a_{iN}b_{iN}]$,
  for $j \leftarrow 1$ to $N-1$ **do**
    $\vec{t} \leftarrow \vec{c_i}$ right shift by 2^j units,
    $[\vec{c_i}] \leftarrow \vec{c_i} + \vec{t}$.
  end for
  $\vec{c_i} \leftarrow \vec{c_i} \times \vec{x}$,
end for
$\vec{c} = c_1 + c_2 + \cdots + c_K$
return $\vec{c}$

E. Extended IForest Anomaly Detection on HE

The cloud server uses a trained Extended IForest model to help IoT devices efficiently perform anomaly detection. A second randomization process is involved during the detection. The second randomization process occurs when IForest randomly selects a segmentation value for node segmentation. Since the segmentation values are randomly selected, it can lead to too much scattering to the leaf nodes of the ITree, thereby reducing the detection correctness. For instance, in some cases the maximum height of the ITree is reached, while the abnormal data have not yet been separated, making the ITree inaccurate to detect anomalies.

For the second randomization process, we improve the splitting strategy in tree construction for IForest. To make the path of abnormal data in the ITree shorter and improve the anomaly score, CryptIF splits only one feature during each splitting. To achieve this, CryptIF randomly selects the maximum or minimum value under the feature for segmentation during each round of lookup. In this step, by using either the **ArrayMax**

Algorithm 3: ITree Construction With Encrypted Matrix.

Input: Encrypted matrix X consists of c rows and m columns, current tree height e, height limit l.
Output: an Encrypted ITree.

if $e \geq l$ **then**
  return leaf.
else
  let Q be a list of attributes in X.
  randomly select an attribute $q \in Q$.
  $B_q \leftarrow X_{column_q}$.
  randomly select max or min.
  if max **then**
    //Recursive construction the right child tree.
    $X_{rchild} \leftarrow$ **ArrayMax** (B_q),
    $SplitAtt \leftarrow q$,
    $SplitValue \leftarrow$**ArrayMax**(B_q),
    $X_{lchild} \leftarrow$ Enc_ITree$(X, e+1, l)$.
  else
    //Recursive construction the left child tree.
    $X_{lchild} \leftarrow$**ArrayMin**(B_q),
    $SplitAtt \leftarrow q$,
    $SplitValue \leftarrow$**ArrayMin**(B_q),
    $X_{rchild} \leftarrow$ Enc_ITree$(X, e+1, l)$.
  end if
end if
return Encrypted ITree.

algorithm or the **ArrayMin** algorithm from the previous subsection, the maximum or minimum value of the ciphertext array can be separated into the left or right child nodes of the ITree. Algorithm 3 details the procedure of the ITree construction over an encrypted matrix through a recursive approach. What's more, each feature can be encrypted by the public key for uploading to the cloud server to form an ITree. Thus, feature streams from multiple IoT devices can be trained collaboratively in the cloud server to consolidate the improved IForest model.

According to the Algorithm 3, CryptIF can train the encrypted IForest by encrypted features streamed from IoT devices. Then the ciphertext will be scanned for anomaly detection in the IForest. Sarpatwar et al. propose to use SIMD property of CKKS to implement classification and regression of ciphertexts under decision trees for outsourcing services [45]. ITree is similar to a decision tree. By integrating the ciphertext comparison algorithm Comp with an encrypted IForest (Isolation Forest), anomalies within ciphertext can be effectively detected in the IForest. To achieve this, we first define the node comparison process for the encrypted ITree. For each data point d, we need to determine whether it belongs to the child node N of a parent node P in the ITree. This determination follows the following steps: if N is the right child of node P, the similarity measure (SM) of node N is computed as:

$$SM(N, SplitValue_N, d) = SM(P, SplitValue_P, d) \times \mathbf{Comp}(d \geq SplitValue_P), \tag{5}$$

if N is the left child of node P, the SM value of node N is computed as:

$$SM(N, SplitValue_N, d) = SM(P, SplitValue_P, d) \times \mathbf{Comp}(d < SplitValue_P), \tag{6}$$

Here, the SM value is an indicator used to assess the affiliation of a data point within the tree. It is approximately 0 or 1 and is determined by both the SM value of the parent node and the result of the comparison operation. If the data point d satisfies the respective splitting condition (either for the left or right subtree), the SM value is retained; otherwise, it gradually diminishes, approaching 0, thereby influencing the final determination of the data point's position in the tree. Through this computation process, we can progressively establish the path of the data point in the encrypted IForest, ultimately enabling anomaly detection in ciphertext.

With the above procedure, CryptIF can calculate the SM value of the ciphertext on each leaf node of the ITree. If the SM is close to 1, it means that the ciphertext falls on this leaf node of ITree. Otherwise, it is 0, which means that the ciphertext does not fall on this leaf node. Each leaf node on ITree records the height value. Thus, CryptIF calculates the ciphertext's average height value in EIForest as shown in (7),

$$avg_h = \frac{1}{n} \times \sum_{n}^{1} \left(\sum_{L_n} SM\left(L_n, d\right) \times v_L \right), \tag{7}$$

where n denotes that there are n ITree in the IForest, L_n denotes the leaf node on the nth ITree, and v_L denotes the height where the leaf node is located on ITree.

By combining the SIMD property of CKKS and (7), we propose an algorithm for batch comparison of all nodes in the encrypted ITree. As shown in Algorithm 4, we simplify the complexity of the operation using parallelized operations, the first k terms of output result vector $\mathcal{L}$ are the SM values corresponding to each leaf node, which is to support to the directly calculation of the specific height of the ciphertext in the encrypted ITree. In Algorithm 4, the $\mathbf{Rotate}(ct, r)$ is to shift the ciphertext vector ct to the right r positions. Besides, the calculated result vectors are all encrypted. Finally, the cloud server calculates the average height avg_h and returns it to the edge node. After the edge node decrypts the encrypted result vector, it can calculate the abnormal score locally and compare it with the abnormal score table to determine whether the data is abnormal. The anomaly score is calculated by (8),

$$s(c) = 2^{-\frac{avg_h}{m(\psi)}}, \tag{8a}$$

$$m(\psi) = 2H(\psi - 1) - (2(\psi - 1)/\psi), \tag{8b}$$

where c is the ciphertext to be analyzed, ψ is the number of samples of IForest, and $H(i)$ is the harmonic number which can be estimated by $\ln i + 0.5772156649$. Unlike previous methods that require multiple communication rounds to retrieve comparison results [44], Algorithm 2 enables the cloud server to generate a consolidated result vector in a single computation, thus substantially minimizing both the communication overhead and the computational workload at the edge node.

Algorithm 4: Result Vector Calculation With ITree and **Comp**.

Input: Encrypted ITrees with n node and k leaves, ciphertext c

Output: the first k entries of the encrypted result vector $\mathcal{L}$ contains the SM value for each leaf node.

1: Suppose $N_1N_2,\ldots N_n$ is the order in which nodes of the ITree are visited in a breadth-first-search traversal, and are not leaf nodes.
2: Suppose $[c,n]=[c,c,\ldots n \quad times\ldots,c]$.
3: //Create new vectors to compare.
4: $T^+ \leftarrow [SplitValue_{N_1},n][0,n][SplitValue_{N_2},n-1]$ $[0,n-1]\cdots[SplitValue_{N_n},1][0,1]$.
5: $T^- \leftarrow [0,n][SplitValue_{N_1},n][0,n-1]$ $[SplitValue_{N_2},n-1]\cdots[0,1][SplitValue_{N_n},1]$.
6: $V^+ \leftarrow [c_{SplitValue_{N_1}},n][0,n][c_{SplitValue_{N_2}},n-1]$ $[0,n-1]\cdots[c_{SplitValue_1},1][0,1]$.
7: $V^- \leftarrow [0,n][c_{SplitValue_{N_1}},n][0,n-1][c_{SplitValue_{N_2}},$ $n-1][0,n-2]\cdots[0,1][c_{SplitValue_1},1]$.
8: $M^+ \leftarrow T^+ + V^+$.
9: $M^- \leftarrow T^+ + V^-$.
10: $\mathcal{L} \leftarrow \mathbf{Comp}(M^+ \geq M^-)$.
11: //Perform iteration comparison.
12: **for** $i \leftarrow 0$ to $\frac{n+1}{2}$ **do**
13: $\quad \mathcal{L} \leftarrow \mathcal{L} *$ **Rotate** $(\mathcal{L}, -(2^i * \frac{n+1}{2}))$.
14: **end for**
15: **return** $\mathcal{L}$.

F. Model Update and Re-Training

In real-world IoT environments, due to changes in device operating conditions, network fluctuations, or the evolution of potential attack patterns, the detection capability of the originally trained model may gradually degrade. Therefore, frequent model re-training is essential to maintain the sustained detection performance of CryptIF. Specifically, during daily operations, IoT devices continuously upload newly generated feature streams that have been processed by feature selection and encrypted. The cloud server dynamically triggers model re-training based on certain strategies. Depending on the application scenarios, two types of strategies are provided: (1) Periodic updates, such as automatically re-training the model once every week; and (2) Performance-driven updates, where re-training is immediately initiated when the detection accuracy, recall, or other performance metrics fall below predefined thresholds. During re-training, the cloud server rebuilds an encrypted Extended Isolation Forest model using encrypted feature data collected over a recent time window (e.g., the past 24 hours or the past week). To avoid affecting the real-time detection capability of IoT devices, the previous detection model remains active during the re-training phase. Once the new model has been trained and validated, a seamless model switching procedure is performed to complete the update, thereby ensuring the continuity and stability of the detection system. It is worth emphasizing that the entire re-training process—including data collection, model construction, and model validation—is carried out over encrypted data, ensuring that user privacy is consistently preserved. Furthermore, thanks to CryptIF's parallelized design and efficient encrypted computation pipeline, the system maintains low communication and computation overhead even under frequent re-training scenarios, making it well-suited for resource-constrained IoT environments.

TABLE II
THE FIVE COMMON PARAMETERS OF CKKS HE FOR MORE COMPREHENSIVE BENCHMARK TESTS

	Polynomial Modulus	Coefficient Modulus	Scale
A	8,192	[40,21,21,40]	2^{21}
B	8,192	[60,40,40,60]	2^{40}
C	8,192	[50,30,30,30,50]	2^{30}
D	8,192	[31,26,26,26,26,26,26,31]	2^{26}
E	8,192	[40,21,21,21,21,21,21,40]	2^{21}

IV. EVALUATION

In this section, we thoroughly evaluate CryptIF from the following perspectives: (1) encryption benchmark of CKKS HE on IoT devices, which is the basis of CryptIF; (2) detection efficacy and performance towards different anomalies. We also compare CryptIF with other state-of-the-art approaches, including both ciphertext-based approaches and plaintext-based approaches (3) theoretical security analysis of CryptIF.

We implemented HE for IoT devices using Microsoft SEAL. To evaluate the computational efficiency of ciphertext generation at different security levels and with different parameters, we tested and recorded the performance of CryptIF on a server with an Intel (R) Core (TM) i7-10700 CPU and 16 GB RAM. In addition, we evaluated CryptIF with different anomaly datasets from public repositories. We conducted comparison experiments on these datasets using different anomaly detection approach (i.e., plaintext-based LOF, plaintext-based IForest, plaintext-based AutoEncoder, TFHE-HistogramBased and BGV-FCM). Unless specified, in all the experiments we used the default values recommended by the authors of IForest, Specifically, the number of isolation trees $t = 100$, where t represents the total number of trees in the EIForest ensemble, and the sub-sampling size $\psi = 256$, where ψ denotes the number of samples randomly selected for constructing each tree. The decision threshold for anomaly detection was set to $threshold = 0.6$, which determines the cutoff score above which a data point is classified as an anomaly.

A. IoT Benchmark for Encryption

In CryptIF, IoT devices are tasked with data collection and the transmission of data to nearby edge nodes. In our experimental setup, IoT devices are simulated through software benchmarks, transmitting raw feature streams to the edge nodes. The edge nodes are realized using Raspberry Pi 4 Model B devices (ARM Cortex-A72, 4 GB RAM), where homomorphic encryption and decryption operations are carried out with the Microsoft SEAL library. In addition, we selected the following five common parameters of CKKS HE for more comprehensive benchmark tests as shown in Table II.

TABLE III
SIZES OF CKKS KEYS GENERATED WITH SYMMETRIC AND ASYMMETRIC HE UNDER USING DIFFERENT PARAMETERS

	Asymmetric encryption key size				
	All	Public key	Secret key	Galois keys	Relin keys
A	464.51 KB	311.07 KB	153.53 KB	21.89 MB	938.07 KB
B	689.56 KB	459.66 KB	230.0 KB	32.04 MB	1.34 MB
C	667.68 KB	446.22 KB	221.56 KB	42.23 MB	1.75 MB
D	810.35 KB	540.89 KB	269.56 KB	88.65 MB	3.71 MB
E	790.3 KB	527.65 KB	262.75 KB	86.87 MB	3.63 MB
	Symmetric encryption key size				
	All	Public key	Secret key	Galois keys	Relin keys
A	154.69 KB	/	154.68 KB	21.88 MB	937.23 KB
B	230.02 KB	/	230.01 KB	32.04 MB	1.34 MB
C	221.56 KB	/	221.55 KB	42.23 MB	1.75 MB
D	269.59 KB	/	269.59 KB	88.65 MB	3.71 MB
E	263.68 KB	/	263.67 KB	86.88 MB	3.63 MB

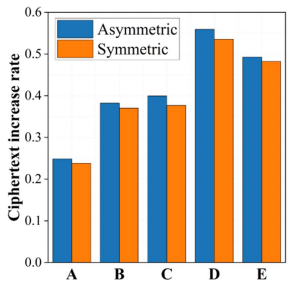


Fig. 5. Ciphertext increase rates in symmetric and asymmetric CKKS HE with different parameters.

In Table III, the size of each key for asymmetric and symmetric HE is listed with five different HE parameter settings. We can see that the size of the key involved in encryption is always maintained at the KB level for both symmetric and asymmetric HE. Moreover, Fig. 5 illustrates the ciphertext increase rate for five different parameter settings. Besides, we can see that the ciphertext increase rates are similar and the increase rates do not exceed 0.5 both with symmetric and asymmetric HE. These can satisfy the low storage and lightweight system design of edge node.

In addition, we performed symmetric and asymmetric HE on a vector consisting of 1000 floating-point numbers. Due to the approximate encryption property, we tested the error range and running times of the ciphertext comparison operation for CryptIF in the cloud service. By using the SIMD property of CKKS HE, CryptIF parallelizes the comparison operations for 1000 chunks of ciphertext. In Tables IV and V, we can see the time consumption of comparison for asymmetric and symmetric HE with different $f(x)$ and $f \circ g(x)$. Moreover, we can see that the error ranges can reach 2^{-22} under specific parameters.

Comparing the while comparison effect of symmetric and asymmetric encryption under different $f(x)$ and $f \circ g(x)$. The results are shown in Tables IV and V. Since CKKS is a floating-point approximation encryption, the table shows the specific error and time of the comparison results and it can be seen that the error accuracy can reach 2^{-22} under specific parameters and has good time efficiency.

In summary, with the above evaluations, we can confirm that CryptIF can be deployed in IoT devices with limited computing power and lightweight systems. Besides, CryptIF operates efficiently on the cloud server side.

B. Correctness and Effectiveness of CryptIF

After the benchmark evaluation in the previous part, we chose $\{8192, [60,40,40,60], 2^{40}\}$ and $f_1^{(8)} \circ g_1^{(7)}$ to conduct the ciphertext anomaly detection. In this subsection, we compare plaintext-based anomaly detection with ciphertext-based anomaly detection. Initially, we utilize eight publicly available anomaly detection datasets, including two network intrusion datasets (i.e., HTTP and SMTP), two IoT network intrusion datasets (i.e., UNSW-NB15 and Ton-IoT network), as well as ForestCover, Shuttle, Pima, and Satellite datasets [14], [46]. The characteristics of these datasets are summarized in Table VI, where n denotes the number of instances, m represents the number of attributes (dimensions), and the anomaly ratio indicates the proportion of anomalous instances. We compare the proposed CryptIF method against conventional plaintext-based anomaly detection approaches—LOF [41], traditional IForest, and traditional Autoencoder—as well as ciphertext-based methods, including TFHE-Histogram-Based Outlier Detection(TFHE-HBOD) [47], EVAD [48] and BGV-FCM [36]. Specifically, EVAD combines the CKKS FHE scheme with the One-Class Support Vector Machine (One-Class SVM) for privacy-preserving anomaly detection. In this section, we report several performance metrics commonly employed in anomaly detection tasks, including Accuracy, ROC-AUC, Recall, False Positive Rate, and F1 score, to comprehensively evaluate the detection models. As listed in Table VII, we can see that the anomaly detection correctness of the CryptIF is not much different from that of the plaintext-based anomaly detection approach (i.e., IForest). What's more, CryptIF significantly outperforms the LOF anomaly detection approach, which is based on plaintexts. Furthermore, in comparison to the deep learning-based AutoEncoder method, CryptIF shows strong competitiveness. While AutoEncoder attains better recall on some small-scale datasets (e.g., Pima), CryptIF achieves a more balanced overall performance in terms of Accuracy and F1 Score. Compared with BGV-FCM and TFHE-HistogramBased, the state-of-art ciphertext-based anomaly detection approach, CryptIF has superior anomaly detection performance due to its better ability to separate anomalies. Compared with EVAD, CryptIF achieves better detection accuracy and efficiency. EVAD combines CKKS with One-Class SVM but is limited by its reliance on polynomial kernels, as nonlinear kernels like RBF or sigmoid are impractical under homomorphic encryption. This restricts its capacity to model complex anomalies. In contrast, CryptIF leverages an EIForest and uses polynomial approximation for ciphertext comparison, effectively bypassing this limitation and improving model expressiveness. Moreover, we plotted the ROC curves for the datasets in Figs. 6 to 13. We can see that CryptIF and IForest have similar detection correctness, and CryptIF consistently outperforms LOF and BGV-FCM across different scenarios.

Then we evaluate the efficiency of CryptIF. In this part, we compare CryptIF with BGV-FCM [36]. Since the BGV HE scheme only suppoerts integer encoding, Alabdulatif et al. proposed to utilize IEEE 754 standard [37] to represent the

TABLE IV
TIME CONSUMPTIONS FOR COMPARING 1,000 PIECES OF CIPHERTEXT ENCRYPTED WITH ASYMMETRIC HE USING SIMD. $(Error\ Range/Runtime\ (s))$.

	$f_1^{(5)}$	$f_4^{(5)}$	$f_4^{(15)}$	$f_1^{(3)} \circ g_1^{(2)}$	$f_1^{(8)} \circ g_1^{(7)}$	$f_2^{(3)} \circ g_2^{(2)}$	$f_2^{(8)} \circ g_2^{(7)}$	$f_3^{(3)} \circ g_3^{(2)}$	$f_3^{(8)} \circ g_3^{(7)}$
A	2^{-2}/0.22350s	2^{-2}/0.38696s	2^{-2}/0.849781s	2^{-2}/0.21343s	2^{-2}/0.43735s	2^{-2}/0.34967s	/	2^{-2}/0.33213s	/
B	2^{-5}/0.21845s	2^{-7}/0.38650s	2^{-22}/0.84927s	2^{-6}/0.21100s	2^{-22}/0.43835s	2^{-8}/0.34964s	2^{-22}/0.74899s	2^{-12}/0.32709s	2^{-22}/0.69511s
C	2^{-5}/0.30077s	2^{-7}/0.53509s	2^{-14}/1.16453s	2^{-6}/0.29919s	2^{-14}/0.58795s	2^{-8}/0.47428s	2^{-14}/0.99533s	2^{-13}/0.45578s	2^{-14}/0.95844s
D	2^{-5}/0.63482s	2^{-8}/1.10369s	2^{-9}/2.34147s	2^{-6}/0.65036s	2^{-9}/1.16704s	2^{-9}/0.95996s	2^{-9}/1.92585s	2^{-10}/0.96245s	2^{-9}/1.89194s
E	2^{-2}/0.63981s	2^{-2}/1.10214s	2^{-2}/2.34942s	2^{-2}/0.65382s	2^{-2}/1.16667s	2^{-2}/0.96648s	2^{-0}/1.97575s	2^{-2}/0.95043s	2^{-0}/1.92884s

TABLE V
TIME CONSUMPTIONS FOR COMPARING 1,000 PIECES OF CIPHERTEXT ENCRYPTED WITH SYMMETRIC HE USING SIMD. $(Error\ Range/Runtime\ (s))$.

	$f_1^{(5)}$	$f_4^{(5)}$	$f_4^{(15)}$	$f_1^{(3)} \circ g_1^{(2)}$	$f_1^{(8)} \circ g_1^{(7)}$	$f_2^{(3)} \circ g_2^{(2)}$	$f_2^{(8)} \circ g_2^{(7)}$	$f_3^{(3)} \circ g_3^{(2)}$	$f_3^{(8)} \circ g_3^{(7)}$
A	2^{-2}/0.19350s	2^{-2}/0.32513s	2^{-2}/0.69167s	2^{-2}/0.24787s	2^{-2}/0.350682s	2^{-2}/0.29077s	/	2^{-2}/0.28922s	/
B	2^{-5}/0.19005s	2^{-7}/0.32567s	2^{-22}/0.69822s	2^{-6}/0.24532s	2^{-22}/0.34806s	2^{-8}/0.28723s	2^{-22}/0.59039s	2^{-12}/0.28327s	2^{-22}/0.56750s
C	2^{-5}/0.27127s	2^{-7}/0.46728s	2^{-14}/0.98146s	2^{-6}/0.33662s	2^{-14}/0.49320s	2^{-8}/0.40843s	2^{-14}/0.81983s	2^{-13}/0.400936s	2^{-14}/0.82379s
D	2^{-5}/0.58695s	2^{-8}/1.00345s	2^{-9}/2.09263s	2^{-6}/0.70418s	2^{-9}/1.02331s	2^{-9}/0.87378s	2^{-10}/1.69347s	2^{-10}/0.887622s	2^{-9}/1.73134s
E	2^{-2}/0.58994s	2^{-2}/1.00737s	2^{-2}/2.10620s	2^{-2}/0.70766s	2^{-2}/1.03367s	2^{-2}/0.8777s	2^{-0}/1.73832s	2^{-2}/0.875627s	2^{-0}/1.7672s

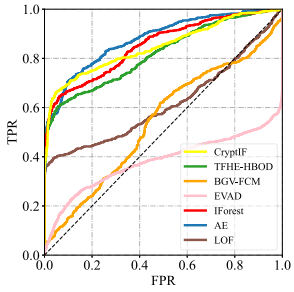


Fig. 6. ROC curves for Satellite anomaly detection (TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based, the others are plaintext-based).

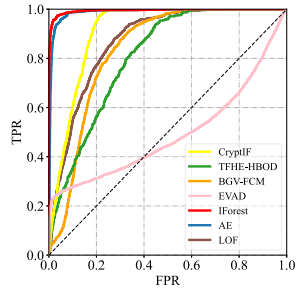


Fig. 9. ROC curves for ForestCover anomaly detection (TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based, the others are plaintext-based).

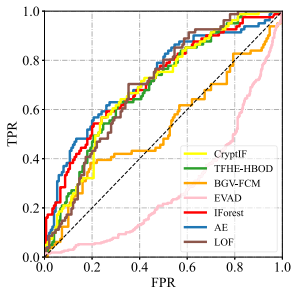


Fig. 7. ROC curves for Pima anomaly detection (TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based, the others are plaintext-based).

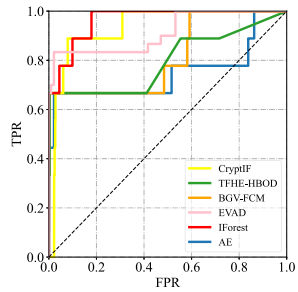


Fig. 10. ROC curves for SMTP anomaly detection (only TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based).

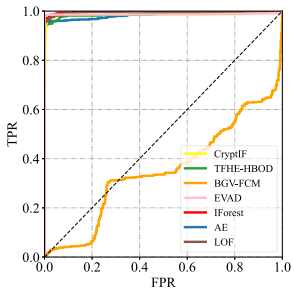


Fig. 8. ROC curves for shuttle anomaly detection (TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based, the others are plaintext-based).

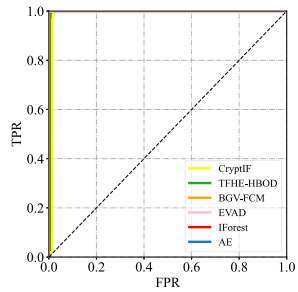


Fig. 11. ROC curves for HTTP anomaly detection (only TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based).

TABLE VI
PROPERTIES OF DATASETS USED IN THIS EVALUATION

Datasets	n	m	Abnormal Proportion
HTTP [14]	567,497	3	0.4%
ForestCover [14]	286,048	10	0.09%
SMTP [14]	95,156	3	0.03%
Shuttle [46]	49,097	9	7%
Satellite [46]	6,435	36	0.32%
Pima [46]	768	8	35%
UNSW-NB15 [49]	2540044	49	63.9%
Ton-IoT network [50]	461045	45	35%

TABLE VII
COMPARISONS OF DETECTION EFFICACY (LOF, IFOREST, AUTOENCODER, TFHE-HBOD, BGV-FCM, AND CRYPTIF)

	Satellite	Pima	Shuttle	ForestCover	HTTP	SMTP	NB15	Ton-IoT
Accuracy								
LOF	0.6812	0.6263	0.8583	0.8068	/	/	0.4743	0.9549
IForest	0.8603	0.6829	0.9689	0.9335	0.9499	0.9690	0.8425	0.8123
AutoEncoder	0.8246	0.6528	0.9646	0.9175	0.95149	0.9498	0.7933	0.6716
TFHE-HBOD	0.7338	0.6926	0.9952	0.9449	0.9006	0.9406	0.5327	0.6208
BGV-FCM	0.7219	0.6627	0.8817	0.9112	0.9169	0.9131	0.7239	0.7385
CryptIF	0.8707	0.6701	0.9692	0.9331	0.9649	0.9711	0.8309	0.8067
EVAD	0.6901	0.6419	0.6470	0.9477	0.8899	0.9867	0.5875	0.5617
ROC-AUC								
LOF	0.5336	0.5009	0.5245	0.5578	/	/	0.4933	0.9774
IForest	0.6627	0.5435	0.9819	0.6851	0.9479	0.9988	0.8825	0.8478
AutoEncoder	0.6812	0.7182	0.9919	0.8353	0.9921	0.8975	0.8321	0.7410
TFHE-HBOD	0.8787	0.6926	0.9864	0.8059	0.9944	0.7963	0.5389	0.5794
BGV-FCM	0.5816	0.5401	0.5825	0.6384	0.9091	0.4591	0.2563	0.8437
CryptIF	0.6434	0.646	0.9779	0.8059	0.9508	0.9989	0.8570	0.8239
EVAD	0.3808	0.2891	0.9867	0.4921	0.9977	0.9410	0.6242	0.4718
Recall								
LOF	0.3895	0.3506	0.1107	0.4211	/	/	0.1342	0.9361
IForest	0.6448	0.5433	0.8435	0.8457	0.8596	0.8159	0.7827	0.7257
AutoEncoder	0.6112	0.5123	1.0	0.8219	1.0	0.7000	0.7600	0.5309
TFHE-HBOD	0.1588	0.1358	0.9364	0.2354	1.0	0.6667	0.1566	0.4854
BGV-FCM	0.4125	0.3741	0.3751	0.7537	0.6837	0.7657	0.6615	0.6266
CryptIF	0.7122	0.5223	0.8571	0.8231	0.8714	0.9351	0.7583	0.7069
EVAD	0.1723	0.1186	0.9857	0.2511	0.9977	0.7124	0.4684	0.4062
False Positive Rate								
LOF	0.7015	0.3840	0.7215	0.6553	/	/	0.0834	0.0350
IForest	0.1961	0.2448	0.1854	0.3854	0.2104	0.1549	0.015	0.1241
AutoEncoder	0.0568	0.0400	0.0489	0.03122	0.0492	0.0497	0.0483	0.2527
TFHE-HBOD	0.0020	0.0067	0.0002	0.0759	0.0997	0.0594	0.0054	0.3064
BGV-FCM	0.3927	0.3126	0.3157	0.4529	0.2841	0.3015	0.1995	0.2013
CryptIF	0.1807	0.2164	0.1573	0.3821	0.1572	0.1254	0.0231	0.1470
EVAD	0.072	0.024	0.3709	0.0453	0.1101	0.0131	0.3746	0.3547
F1 Score								
LOF	0.2071	0.1381	0.1199	0.4007	/	/	0.1919	0.9356
IForest	0.6448	0.5433	0.8189	0.6611	0.6751	0.7315	0.8740	0.5306
AutoEncoder	0.6213	0.5109	0.9402	0.6034	0.5839	0.7009	0.7932	0.8547
TFHE-HBOD	0.2740	0.2366	0.9657	0.0759	0.0727	0.0070	0.2891	0.4723
BGV-FCM	0.3121	0.5401	0.5825	0.6826	0.501	0.6983	0.7252	0.6266
CryptIF	0.6430	0.5462	0.8762	0.6434	0.6540	0.7044	0.8550	0.7010
EVAD	0.2603	0.1335	0.2854	0.2845	0.1659	0.0328	0.3527	0.3931

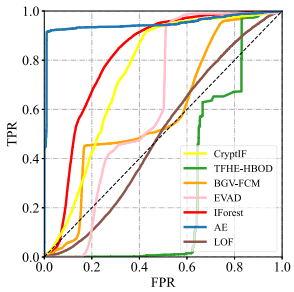


Fig. 12. ROC curves for UNSW_NB15 anomaly detection (TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based, the others are plaintext-based).

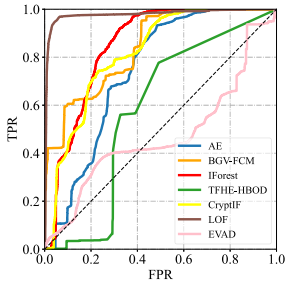
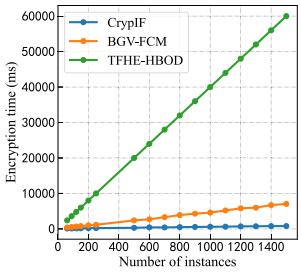


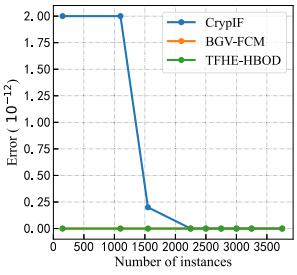
Fig. 13. ROC curves for Ton-IoT anomaly detection (TFHE-HBOD, BGV-FCM, EVAD and CryptIF are ciphertext-based, the others are plaintext-based).

floating-point number. As illustrated in Fig. 14(a), we can see that BGV-FCM requires more encryption time than CryptIF because of using 32-bit binary to represent the floating-point numbers. Therefore, CryptIF is far better than BGV-FCM regarding the encryption time and power consumption on the IoT side, which means CryptIF is more easily deployable. On the other hand, CKKS HE is an approximate HE, so it will also bring errors in the training phase. Fig. 14(b) illustrates that with the increase of the number of instances, the error will become close to 0 due to the large-scale sample size. Moreover, in Fig. 14(c), we can see that CryptIF can reach convergence in time due to the number of samples and ITrees in EIForest when training the detection model. In contrast, the FCM clustering algorithm increases the running time continuously with the increasing number of instances. Therefore, the anomaly detection algorithm for encrypted data proposed in this paper outperforms the BGV-FCM approach in terms of efficiency. Finally, we test the detection time consumption for CryptIF with different parameter settings. BGV-FCM does not engage in this evaluation because it can perform anomaly detection during the training phase. As illustrated in Fig. 14(d), we can see that the running time grows linearly with the increasing number of instances. Because by using SIMD, CryptIF can reduce the n times comparisons to $\frac{n+1}{2}$ times in EIForest. Although EVAD uses CKKS, which supports SIMD, the structure of One-Class SVM with polynomial kernels limits the extent to which parallelization can be effectively applied. Kernel matrix evaluations require encrypted inner product operations between individual data points, which involve multiple rotations and rescalings, reducing SIMD efficiency. In contrast, CryptIF's tree-based structure and comparison-by-approximation design are inherently more parallelizable and scalable. Moreover, each ITree detection task can be performed in parallel in the cloud server, so the time taken by CryptIF to detect a ciphertext is equivalent to the time taken by an encrypted ITree to detect a ciphertext. In addition to the comparison with the BGV-FCM method, we further extend our evaluation by including the TFHE-Histogram approach.

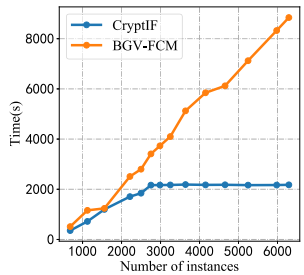
Experimental results indicate that the TFHE-Histogram method incurs substantially higher encryption overhead compared to CryptIF, primarily due to its reliance on bit-level homomorphic encryption. Moreover, the coarse granularity inherent in histogram-based detection limits its ability to accurately identify anomalies near decision boundaries, leading to relatively



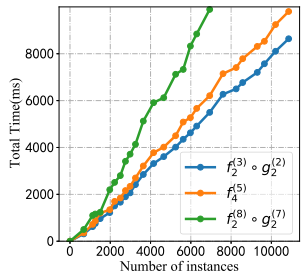
(a) Encryption time comparison. Due to the limitation that only integers can be encrypted, BGV-FCM uses IEEE 754 to encrypt floating point numbers, which requires significantly more encryption time than CKKS. TFHE encrypts data bit by bit, leading to the highest encryption time.



(b) Error rates. Due to the approximately HE property, CKKS may cause detection errors on smaller scale data. But when the number of instances is large enough, the error can be reduced to approximately zero.



(c) Training time comparison (CryptIF vs. BGV-FCM). CryptIF takes significantly less time to train the model especially with a large number of instances. HBOD is excluded from the comparison as it is a training-free method. EVAD is also excluded, as its model is trained in plaintext and thus not comparable in the encrypted training setting.



(d) Detection time of CryptIF with different parameter settings. In all cases, the detection time increases linearly with the number of instances, which is helpful for high-performance cloud server to handle large-scale anomaly detection.

Fig. 14. Performance comparisons (TFHE-HBOD versus BGV-FCM versus CryptIF and between different parameter settings for CryptIF).

higher detection errors. Although this approach avoids the need for a training convergence phase, its dependence on intensive ciphertext-level bitwise comparisons significantly increases detection time in large-scale scenarios.

Overall, CryptIF outperforms the EVAD, TFHE-HBOD, and BGV-FCM methods by offering improved encryption efficiency, higher detection accuracy, and lower inference latency. Furthermore, CryptIF demonstrates operating performance that is nearly indistinguishable from its plaintext counterpart, IForest. In terms of processing time, the construction time of IForest consistently converges to a stable value regardless of the dataset size, while CryptIF's detection time scales linearly with the number of instances. These results confirm that CryptIF is both effective for anomaly detection and capable of preserving data privacy, making it well-suited for secure IoT applications.

C. Security Analysis

In this subsection we analyze the security of CryptIF. First, we assume that the cloud server is in a semi-honest model, which means it is curious about users but will not perform malicious operations. Second, the selected CKKS HE in CryptIF satisfies semantic security. Besides, The security goal of CryptIF is that the cloud server or the attacker with eavesdropping capability can not obtain information about the IoT devices, and can not deduce the raw data through the CryptIF detection procedure. Under the above assumptions, several theorems are given and proved to demonstrate the security of our approach.

Theorem 1: The probability that the cloud server or attackers with eavesdropping capabilities can successfully recover the raw data based on the information they obtained is negligible.

Proof 1: The cloud server or attackers with eavesdropping capability can obtain the information including encrypted feature streams X and the ciphertext d to be detected. All data are encrypted through CKKS HE, and the probability that the cloud server or attackers can successfully obtain the raw data without knowing the private key is equivalent to breaking CKKS. According to the semantic security of HE, the probability of breaking CKKS is negligible. Moreover, as the data transmitted to the cloud server has been processed by the feature selection algorithm, the raw data is impossible to be completely leaked. In the anomaly detection process, the data that can be obtained is the ciphertext data d, and similarly, these data are ciphertexts obtained through CKKS HE. The probability that the cloud server or the attackers successfully obtain the raw data and the detection result without knowing the private key is equivalent to breaking the CKKS.

Theorem 2: The probability that the cloud server or attackers with eavesdropping capabilities will succeed in obtaining the raw data is negligible after obtaining the CryptIF details.

Proof 2: The cloud server or attackers with eavesdropping capabilities can access the specifics of what is executed in CryptIF after obtaining the CryptIF operation details. CryptIF uses **ArrayMax/ArrayMin** and **Comp** for ciphertext operations. In these processes, the computation results are in ciphertexts. The probability that the cloud server or attackers successfully obtain the computation results without knowing the private key is equivalent to breaking CKKS. Due to the SIMD property of CKKS, the features are encrypted in vectors. Therefore, the probability that the cloud server or attackers successfully tamper and steal data from the encrypted vectors without knowing the private key is equivalent to breaking CKKS. Moreover, the CKKS is approximate HE, which is is equivalent to adding noise points to the features. In summary, it is difficult for the cloud server or attackers to recover the raw data even from the data with noise.

V. Conclusion

With the proliferation of IoT, IoT-oriented attacks are becoming far more sophisticated and threatening than before. To break the computing power limits of IoT devices and provide them with comprehensive protections, we propose CryptIF, a

cloud-based IoT anomaly detection approach over encrypted feature streams. It can simultaneously preserve user privacy, achieve decent detection efficacy, and maintain high operating efficiency.

CryptIF leverages FHE to communicate with IoT devices and conduct anomaly detection in a privacy-preserving manner. To increase the detection efficacy and reduce the training time, CryptIF adopts an EIForest algorithm to select the split node through its dissimilitude degree rather than randomly. We also take multiple measures to increase CryptIF's efficiency on both the IoT side and the server side, including selecting high-quality features through statistical approaches and parallelizing the detection tasks with the SIMD property of CKKS HE.

The evaluation results have demonstrated that the efficiency and efficacy of CryptIF are both high. It can outperform the state-of-the-art ciphertext-based anomaly detection approach in terms of detection efficacy. It can also achieve the line speed in message encryption, transmission, and processing.

REFERENCES

[1] Y. Feng, J. Xu, and L. Weymouth, "University blockchain research initiative (UBRI): Boosting blockchain education and research," *IEEE Potentials*, vol. 41, no. 6, pp. 19–25, Nov./Dec. 2022.

[2] State of IoT 2023: Number of connected IoT devices growing 16% to 16.0 billion globally, 2023. Accessed: Aug. 08, 2023. [Online]. Available: https://iotbusinessnews.com/2023/05/25/34645-state-of-iot-2023-number-of-connected-iot-devices-growing-16-to-16-0-billion-globally-wi-fi-bluetooth-and-cellular-driving-the-market/

[3] Alto Palo Networks, "2020 unit 42 IoT threat report," 2022. Accessed: Jul. 25, 2022. [Online]. Available: https://start.paloaltonetworks.com/unit-42-iot-threat-report

[4] M. Antonakakis et al., "Understanding the mirai botnet," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 1093–1110.

[5] V. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *Proc. IEEE Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber Phys. Social Comput. IEEE Smart Data*, 2016, pp. 164–170.

[6] S. Gandrén and N. Berild Lundblad, "Smart household devices and the negotiation of public and private spaces," *Faculty Law*, Stockholm University Research Paper Series, no. 94, 2021, doi: 10.2139/ssrn.3902051.

[7] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Netw.*, vol. 11, no. 8, pp. 2661–2674, 2013.

[8] M. Gaglianese, S. Forti, F. Paganelli, and A. Brogi, "Lightweight self-adaptive cloud-IoT monitoring across Fed4Fire testbeds," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2022, pp. 1–6.

[9] J. Sun, Y. Yuan, M. Tang, X. Cheng, X. Nie, and M. U. Aftab, "Privacy-preserving bilateral fine-grained access control for cloud-enabled industrial IoT healthcare," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6483–6493, Sep. 2022.

[10] B. Jiang, J. Li, G. Yue, and H. Song, "Differential privacy for industrial Internet of Things: Opportunities, applications, and challenges," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10430–10451, Jul. 2021.

[11] H. Goyal and S. Saha, "Multi-party computation in IoT for privacy-preservation," 2022, *arXiv:2206.01956*.

[12] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[13] Q. Pan, J. Wu, X. Zheng, W. Yang, and J. Li, "Differential privacy and irs empowered intelligent energy harvesting for 6G Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22109–22122, Nov. 2022.

[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[15] J. H. Cheon, D. Kim, and D. Kim, "Efficient homomorphic comparison methods with optimal complexity," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2020, pp. 221–256.

[16] C. Gu, X. Cui, X. Zhu, and D. Hu, "FL2DP: Privacy-preserving federated learning via differential privacy for artificial IoT," *IEEE Trans. Ind. Informat.*, vol. 20, no. 4, pp. 5100–5111, Apr. 2024.

[17] K. Sahinbas and F. O. Catak, "Secure multi-party computation-based privacy-preserving data analysis in healthcare IoT systems," in *Interpretable Cognitive Internet of Things for Healthcare*, Berlin, Germany: Springer, 2023, pp. 57–72.

[18] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2017, pp. 409–437.

[19] M. Yagisawa, "Fully homomorphic encryption without bootstrapping," *IACR Cryptolo. ePrint Arch.*, vol. 2015, no. 474, 2015.

[20] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," *Cryptol. ePrint Arch.*, Paper 2012/144, 2012. [Online]. Available: https://eprint.iacr.org/2012/144

[21] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, 2020.

[22] A. Aloufi, P. Hu, H. W. Wong, and S. S. Chow, "Blindfolded evaluation of random forests with multi-key homomorphic encryption," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1821–1835, Jul./Aug. 2021.

[23] J. H. Cheon, D. Kim, D. Kim, H. H. Lee, and K. Lee, "Numerical method for comparison on homomorphically encrypted numbers," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Springer, 2019, pp. 415–445.

[24] S. Hong, S. Kim, J. Choi, Y. Lee, and J. H. Cheon, "Efficient sorting of homomorphic encrypted data with k-way sorting network," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4389–4404, 2021.

[25] D. Chen et al., "Privacy-preserving encrypted traffic inspection with symmetric cryptographic techniques in IoT," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17265–17279, Sep. 2022.

[26] K. Kethineni and P. Gera, "IoT-based privacy-preserving anomaly detection model for smart agriculture," *Systems*, vol. 11, no. 6, p. 304, 2023.

[27] R. Shrestha et al., "Anomaly detection based on LSTM and autoencoders using federated learning in smart electric grid," *J. Parallel Distrib. Comput.*, vol. 193, 2024, Art. no. 104951.

[28] D. Chen et al., "Privacy-preserving anomaly detection of encrypted smart contract for blockchain-based data trading," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4510–4525, Sep./Oct. 2024.

[29] Y. Feng et al., "Unmasking the internet: A survey of fine-grained network traffic analysis," *IEEE Commun. Surveys Tuts.*, early access, Feb. 25, 2025, doi: 10.1109/COMST.2025.3545541.

[30] Y. Feng, J. Li, and D. Sisodia, "CJ-Sniffer: Measurement and content-agnostic detection of cryptojacking traffic," in *Proc. 25th Int. Symp. Res. Attacks Intrusions Defenses*, New York, NY, USA, 2022, pp. 482–494, doi: 10.1145/3545948.3545973.

[31] Y. Feng, J. Luo, C. Ma, T. Li, and L. Hui, "I can still observe you: Flow-level behavior fingerprinting for online social network," in *Proc. IEEE Glob. Commun. Conf.*, 2022, pp. 6427–6432.

[32] Y. Feng, J. Li, L. Jiao, and X. Wu, "Towards learning-based, content-agnostic detection of social bot traffic," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2149–2163, Sep./Oct. 2021.

[33] Y. Feng, J. Li, L. Jiao, and X. Wu, "BotFlowMon: Learning-based, content-agnostic identification of social bot traffic flows," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2019, pp. 169–177.

[34] A. Alabdulatif, H. Kumarage, I. Khalil, and X. Yi, "Privacy-preserving anomaly detection in cloud with lightweight homomorphic encryption," *J. Comput. Syst. Sci.*, vol. 90, pp. 28–45, 2017.

[35] J. H. Cheon, W.-H. Kim, and H. S. Nam, "Known-plaintext cryptanalysis of the Domingo-Ferrer algebraic privacy homomorphism scheme," *Inf. Process. Lett.*, vol. 97, no. 3, pp. 118–123, 2006.

[36] A. Alabdulatif, I. Khalil, H. Kumarage, A. Y. Zomaya, and X. Yi, "Privacy-preserving anomaly detection in the cloud for quality assured decision-making in smart cities," *J. Parallel Distrib. Comput.*, vol. 127, pp. 209–223, 2019.

[37] W. Kahan, "IEEE standard 754 for binary floating-point arithmetic," *Lecture Notes Status IEEE*, vol. 754, no. 94720/1776, 1996, Art. no. 11.

[38] M. Du, R. Jia, and D. Song, "Robust anomaly detection and backdoor attack detection via differential privacy," 2019, *arXiv: 1911.07116*.

[39] D. Li, X. Liao, T. Xiang, J. Wu, and J. Le, "Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation," *Comput. Secur.*, vol. 90, 2020, Art. no. 101701.

[40] K. Itokazu, L. Wang, and S. Ozawa, "Outlier detection by privacy-preserving ensemble decision tree u sing homomorphic encryption," in *Proc. 2021 Int. Joint Conf. Neural Netw.*, 2021, pp. 1–7.

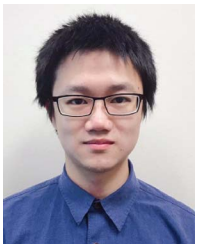
[41] M. N. Kurt, Y. Yılmaz, X. Wang, and P. J. Mosterman, "Online privacy-preserving data-driven network anomaly detection," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 982–998, Mar. 2022.

[42] M. Arazzi, S. Nicolazzo, and A. Nocera, "A fully privacy-preserving solution for anomaly detection in IoT using federated learning and homomorphic encryption," *Inf. Syst. Front.*, vol. 27, pp. 367–390, 2025.

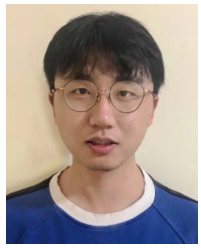
[43] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1479–1489, Apr. 2021.
[44] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," 2021, *arXiv:2104.03152*.
[45] K. Sarpatwar et al., "Privacy enhanced decision tree inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 34–35.
[46] K. Ting, S. Tan, and F. Liu, "Mass: A new ranking measure for anomaly detection," *IEEE Trans. Knowl. Data Eng.*, pp. 1–13, 2009.
[47] A. Hangan, D. Lazea, and T. Cioara, "Privacy preserving anomaly detection on homomorphic encrypted data from IoT sensors," 2024, *arXiv:2403.09322*.
[48] A. Falcetta and M. Roveri, "EVAD: Encrypted vibrational anomaly detection with homomorphic encryption," *Neural Comput. Appl.*, vol. 36, no. 13, pp. 7359–7372, 2024.
[49] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. 2015 Mil. Commun. Inf. Syst. Conf.*, 2015, pp. 1–6.
[50] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165130–165150, 2020.

Teng Li received the BS degree from the School of Computer Science and Technology, Xidian University, China, in 2013, and the PhD degree from the School of Computer Science and Technology, Xidian University, China, in 2018. He is currently a professor with the School of Cyber Engineering, Xidian University, China. His current research interests include wireless and mobile networks, distributed systems and intelligent terminals with focus on security and privacy issues.

Zejian Lin received the BE degree from the School of Cyber Engineering, Xidian University, Xi'an, China, in 2024, where he is currently working toward the master's degree with the School of Cyber Engineering. His main research interests include fault detection, deep learning,anomaly detection.



Yebo Feng received the PhD degree in computer science from the University of Oregon (UO), in 2023. He is a research fellow with the College of Computing and Data Science (CCDS), Nanyang Technological University (NTU). His research interests include network security, blockchain security, and anomaly detection. He is the recipient of the Best Paper Award of 2019 IEEE CNS, Gurdeep Pall Graduate Student Fellowship of UO, and Ripple Research Fellowship. He has served as the reviewer of *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *ACM Transactions on Knowledge Discovery from Data*, *IEEE Journal on Selected Areas in Communications*, *IEEE Communications Surveys and Tutorials*, etc. Furthermore, he has been a member of the program committees for international conferences including SDM, CIKM, and CYBER, and has also served on the Artifact Evaluation (AE) committees for USENIX OSDI and USENIX ATC.



Chong Wang received the bachelor's and PhD degrees from Fudan University, in 2018. He is currently a research fellow with Cyber Security Laboratory, Nanyang Technologicai University. His research interests lie in the crossroads between artificial intelligence and software engineering. His overarching research mission revolves around enhancing both productivity and security within the realm of software development. He has published multiple papers in international journals and conferences, such as *IEEE Transactions on Software Engineering (TSE)*, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, ACM Symposium on the Foundations of Software Engineering (FSE), IEEE/ACM International Conference on Automated Software Engineering (ASE) and IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).

Zhuo Ma (Senior Member, IEEE) received the PhD degree in computer architecture from Xidian University, Xi'an, China, in 2010. He is currently a professor with the School of Cyber Engineering, Xidian University. His research interests include cryptography, machine learning in cyber security, and Internet of Things security.

Bin Xiao received the BS and MS degrees in electrical engineering from Shanxi Normal University, Xi'an, China, in 2004 and 2007, respectively, and the PhD degree in computer science from Xidian University, Xi'an. He is currently a professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include image processing and pattern recognition.

Jianfeng Ma (Member, IEEE) received the PhD degree from Xidian University, Xi'an, China, in 1995. He has been a professor with the Department of Computer Science and Technology, Xidian University since 1998. He was a Special Engaged professor of the Yangtze River Scholar, China. His research interests include cryptology, network security, and data security.

Yang Liu (Senior Member, IEEE) is currently a full professor and the director of the Cyber Security Lab, Nanyang Technological University, Singapore. He specializes in software security, verification, software engineering and artificial intelligence. His research has bridged the gap between the theory and practical usage of formal methods and program analysis to evaluate the design and implementation of software for high assurance and security. His work led to the development of state-of-the-art model checker, Process Analysis Toolkit (PAT). By now, he has more than 200 publications and 6 best paper awards in top tier conferences and journals. With more than 50 million Singapore dollar funding support, he is leading a large research team working on state-of-the-art software engineering and cyber security problems and currently serving as an associated editor of *IEEE Transactions on Information Forensics and Security*.