Deduplication and approximate analytics for encrypted IoT data in fog-assisted cloud storage

Rongxi Wang[1,2], Guanxiong Ha[1,2], Chunfu Jia[1,2,*], Ruiqi Li[3], and Zhen Su[1,2]

[1] The College of Cyber Science, Nankai University, Tianjin, China.
[2] Tianjin Key Laboratory of Network and Data Security Technology, Tianjin, China.
[3] The College of Safety Science and Engineering, Civil Aviation University of China, Tianjin, China.

Abstract. With the advancement of the Internet of Things (IoT) technologies, there has been a rapid increase in the volume of IoT data, leading to escalating costs in storage, transmission, and analytics. The benefits of conventional data deduplication schemes are diminishing when applied to IoT data that is similar but distinct, necessitating the development of new approaches to accommodate these new scenarios. This paper proposes a deduplication and approximate analytics scheme for encrypted IoT data in fog-assisted cloud storage. The scheme is based on Generalized Deduplication(GD), Message Locked Encryption(MLE), Homomorphic Encryption(HE), and ciphertext conversion techniques. We employ GD to divide similar but distinct IoT data into bases and deviations, and perform deduplication on the encrypted base to achieve efficient storage while protecting data privacy. Additionally, we utilize Hybrid Homomorphic Encryption(HHE) techniques to convert the symmetric ciphertext of IoT data into homomorphic ciphertext, facilitating approximate analytics while ensuring privacy protection of IoT data in fog-assisted cloud storage and reducing the computation overhead on IoT devices.

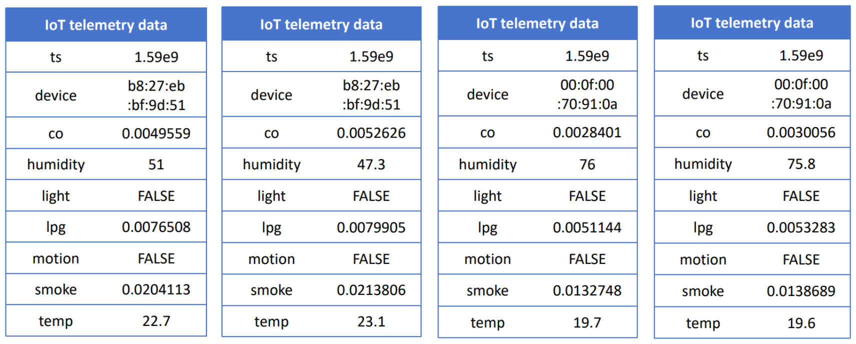
Keywords: Fog-assisted cloud storage, encrypted deduplication, IoT data, homomorphic encryption, approximate analytics

1 Introduction

In recent years, as IoT technology has developed, IoT devices have gained more widespread application [21,24]. However, with the extensive proliferation of IoT devices, a massive volume of data has also emerged. This abundance of IoT data poses numerous challenges for IoT systems, such as increased communication overhead between devices and servers, greater storage costs, and higher data transmission delays and analytics times. These issues have led to the adoption of two solutions: iterative storage and transfer storage [9]. Iterative storage can

This work was supported by National Key R&D Program of China (2018YFA0704703) and National Natural Science Foundation of China (62172238).
* Corresponding Author. E-mail: cfjia@nankai.edu.cn

cause problems such as difficulty in data traceability and compromised data integrity [31]. Transfer storage has introduced fog storage [30], which can reduce latency and bandwidth consumption, alleviating the pressure on devices and servers. In light of these challenges, many companies, such as Huawei and AT&T , have employed fog-assisted cloud storage architectures to manage IoT data [27, 33].



IoT telemetry data	
ts	1.59e9
device	b8:27:eb :bf:9d:51
co	0.0049559
humidity	51
light	FALSE
lpg	0.0076508
motion	FALSE
smoke	0.0204113
temp	22.7

IoT telemetry data	
ts	1.59e9
device	b8:27:eb :bf:9d:51
co	0.0052626
humidity	47.3
light	FALSE
lpg	0.0079905
motion	FALSE
smoke	0.0213806
temp	23.1

IoT telemetry data	
ts	1.59e9
device	00:0f:00 :70:91:0a
co	0.0028401
humidity	76
light	FALSE
lpg	0.0051144
motion	FALSE
smoke	0.0132748
temp	19.7

IoT telemetry data	
ts	1.59e9
device	00:0f:00 :70:91:0a
co	0.0030056
humidity	75.8
light	FALSE
lpg	0.0053283
motion	FALSE
smoke	0.0138689
temp	19.6

Fig. 1: Environmental monitoring IoT data.

As illustrated in Fig. 1, each IoT data record primarily consists of two parts. One part is the data monitored by the device, such as carbon monoxide and liquid petroleum gas; the other part is information about the device, such as the device's name or number. Most IoT data are highly similar but distinct, primarily because the devices generating the data are of the same type. The minor variations in highly similar data usually stem from differences in geographical location or time, or both. Additionally, the inherent periodicity of IoT device operations results in significant redundancy within IoT data [8]. Regarding storage, the large volume of duplicate content in IoT data makes data deduplication and compression essential strategies for reducing storage overhead. Deduplication techniques, which use data blocks of several kilobytes as units for redundancy elimination, can efficiently remove duplicate data to enhance storage performance. However, conventional deduplication schemes can only detect and remove identical data blocks, not those that are highly similar but distinct. Compression is divided into lossy compression, which can result in data loss, and lossless compression, which typically involves considerable computational effort and is challenging to implement on resource-constrained IoT devices.

Considering that conventional data deduplication and compression schemes are no longer applicable and given the characteristics of IoT data, Vestergaard *et*

https://www.huawei.com/en
https://www.att.com/
https://www.kaggle.com/datasets/garystafford/environmental-sensor-data-132k/data

al. [29] proposed Generalized Deduplication(GD) to eliminate redundant parts in data that are similar but distinct. This is a lossless compression scheme. This approach uses a transformation function to separate data into bases and deviations, which has been effective for IoT data [28]. GD is based on heuristic algorithms that generate permutation and transformation functions to process IoT data. The base includes the same bits among different data, while the deviations are the different bits. Since the base is consistent and the high similarity of IoT data introduces substantial redundancy within it, deduplicating the bases can significantly reduce storage overhead. Additionally, data processed through GD retains randomly accessible features and can undergo approximate analytics without decompression [28]. Furthermore, since IoT data often contains a large amount of sensitive information, such as health data and identity information from some devices, directly uploading plaintext data to fog nodes and cloud servers can easily lead to privacy breaches. Therefore, IoT data is typically encrypted before being uploaded.

However, data encryption can affect deduplication and computational analysis. Encrypted data cannot be processed for deduplication. *How to perform deduplication on encrypted data is an important issue.* And, IoT data holds significant value for data analytics, such as in medical health diagnostics or environmental monitoring. Conventional encryption schemes render the data non-computable after encryption. A key issue is *how to make IoT data computable while ensuring data privacy.* Moreover, IoT data contains a large amount of redundant content. By deduplicating the data before performing approximate analytics, we can achieve results that, while not perfectly accurate, are usually close enough to the true values to be practical and valuable for real-world applications. In terms of IoT data analytics, existing solutions perform approximate analytics on plaintext, which can lead to privacy concerns [14]. *How to reduce the computation overhead on IoT devices and the communication overhead between IoT devices and fog nodes is also a critical issue.* Although Homomorphic Encryption(HE) allows for approximate analytics on ciphertexts, directly using it significantly increases the computation burden on IoT devices and results in substantial communication overhead. We will explore methods for deduplication and approximate analytics of encrypted IoT data.

1.1 Contribution

In this paper, we propose a deduplication and approximate analytics scheme for encrypted IoT data in fog-assisted cloud storage. We combine GD and Message Locked Encryption(MLE) to achieve encrypted deduplication for similar but distinct IoT data in fog-cloud storage, enhancing storage performance while protecting data privacy. Specifically, we use GD to divide data into bases and deviations. Given the large volume and high redundancy of the bases, we perform MLE-based encrypted deduplication on it to save storage overhead. Since the deviations occupy little data and have a low duplication rate, we do not perform deduplication on the deviations. Additionally, we introduce Hybrid Homomorphic Encryption(HHE) to enable approximate analytics of encrypted IoT data using K-modes in fog-assisted cloud storage. To save computation and communication overhead, we perform MLE on bases and HE on the relatively small symmetric encryption keys. Devices send the encrypted bases and encrypted keys to fog nodes, and the encrypted deviations to the cloud server. Approximate analytics is primarily conducted at the fog nodes. During approximate analytics, the fog node first re-encrypts the symmetric ciphertext into homomorphic ciphertext. Then, it uses the homomorphically encrypted MLE keys to perform symmetric decryption on homomorphic ciphertext. This ensures that the conversion from symmetric ciphertext to homomorphic ciphertext protects data privacy without exposing the plaintext. After conversion to homomorphic ciphertext, the fog nodes can conduct approximate analytics on it. In summary, we make the following contributions:

1. We combine GD and MLE to achieve encrypted deduplication of similar but distinct IoT data on fog nodes, providing privacy protection and efficient storage.
2. We employ a ciphertext transformation technique to facilitate approximate analytics on encrypted IoT data, which also reduces the computation overhead for IoT devices. During the approximate analytics process, we use HHE to convert symmetric ciphertext into homomorphic ciphertext. This allows for the direct execution of K-modes clustering operations on the ciphertext, enabling approximate analytics.
3. We conduct a security analysis of our scheme, demonstrating its security. Additionally, compared to non-deduplicated data, our scheme reduces storage overhead by approximately 85% through encrypted deduplication. It also employs HHE, which decreases the communication overhead between IoT devices and fog nodes by about 95%, and reduces the time for IoT devices to encrypt data by around 88%. Moreover, our scheme also achieves approximate analytics of homomorphically encrypted IoT data with acceptable efficiency.

1.2 Paper organization

The remainder of this paper is organized as follows. Section 2 discusses the background and motivation of our work. Section 3 describes the preliminary knowledge relevant to our study. In Section 4, we provide an overview of our scheme, including the system architecture and threat model. Section 5 is dedicated to a security analytics of our scheme. In Section 6, we evaluate the performance of our scheme. Finally, we conclude the paper in Section 7.

2 Background and motivation

This section discusses conventional encrypted deduplication, as well as more suitable GD schemes for IoT data. This section also covers approximate analytics. The security and limitations of existing schemes serve as the motivation for our research work.

2.1 Conventional encrypted deduplication

Data deduplication, a technique that retains only one copy of original data and uses pointers or references for identical data, is highly effective for identifying and eliminating redundancy, thus significantly enhancing system storage efficiency. Encrypted deduplication combines encryption and deduplication technologies to achieve efficient data storage while protecting data privacy. Convergent encryption (CE) is the first scheme to implement encrypted deduplication [5]. CE generates a key based on the plaintext content, thus identical plaintexts correspond to the same ciphertext, achieving the purpose of encrypted deduplication. Subsequently, Bellare *et al.* [2] formalizes CE as MLE and provided a security definition for it.

However, as mentioned in [16], MLE is susceptible to offline brute-force attacks(BFAs) [16] when the plaintext entropy is low. An adversary can enumerate all possible plaintexts and compare them with corresponding ciphertexts to guess the correct plaintext message. Considering this issue, Keelveedhi *et al.* [16] proposes server-aided MLE, where the key is derived from both the message itself and a key server. As long as adversaries cannot access the key server, they are unable to perform offline BFAs. Additionally, the key server can perform rate limiting to resist online BFAs. Subsequent schemes [11, 18, 19, 22, 26] are based on server-aided MLE. However, these encrypted deduplication schemes are not suitable for fog-assisted cloud storage architectures and are incapable of handling similar but distinct IoT data.

2.2 Generalized deduplication

Data compression is categorized into lossy and lossless compression. While lossy compression can achieve higher compression rates, it risks data loss. On the other hand, conventional lossless compression schemes, such as Zstd [6] and bzip2 [25], become less effective when applied to smaller data blocks. Therefore, conventional data deduplication and compression techniques are not suitable for IoT data, necessitating new technologies to improve the storage and analytics efficiency of IoT data.

GD [29] has developed a new approach based on the principles of data compression and deduplication to handle IoT data, which often consists of similar but distinct entries. The main technique in GD is using a transformation function to separate data into bases and deviations. The main process of GD is illustrated in Fig. 2.

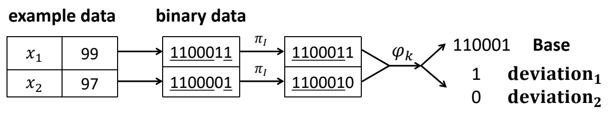


Fig. 2: The main process of generalized deduplication.

In our scheme, we employ the GD algorithm at the device end to process IoT data, obtaining the data's bases and deviations. Given the high redundancy in the bases and low redundancy in the deviations, we perform deduplication on the bases while retaining all deviations.

2.3 Approximate analytics

Approximate analytics comprises a range of computational techniques that do not guarantee precise results but are sufficient for practical applications [23]. The underlying idea of approximate analytics is that while precise analytics is possible, it often consumes excessive resources. Approximate analytics, on the other hand, can significantly conserve resources while delivering results within an acceptable range. For instance, in the case of the K-means clustering algorithm, accepting a 5% accuracy loss can save about 50 times the resources compared to a precise analytics.

Clustering algorithms are a type of unsupervised learning algorithm in machine learning. Without labeled information, clustering algorithms identify the inherent patterns in data and group it into multiple classes or clusters. Each cluster or class contains data that are highly similar to each other but distinctly different from data in other clusters.

Currently, many schemes [7,32] use HE for machine learning on different data sets, but no schemes exist for machine learning on homomorphically encrypted similar but distinct IoT data. Additionally, there are no schemes for machine learning on encrypted IoT data within fog-assisted cloud storage. Given the characteristics of IoT data, which are similar but distinct and contain a lot of redundancy, approximate analytics is more suitable.

In our scheme, we use the K-modes clustering algorithm to perform approximate analytics on similar but distinct IoT data.

3 Preliminaries

This section provides an overview of HE, MLE, Fasta, and K-modes.

3.1 Homomorphic encryption

As noted by Craig Gentry, the first scholar to construct a FHE scheme in [10], HE is a method that allows data processing to be delegated without disclosing access rights to the data. The FHE scheme primarily includes the following algorithms:

- HE.KeyGen(λ): It takes a security parameter λ as input and outputs a public key pk and a private key sk.
- HE.Enc(m, pk): It takes a public key pk and a plaintext message m as input and outputs the ciphertext c of m.
- HE.Dec(c, sk): It takes a private key sk and a ciphertext c of m as input and outputs the plaintext m.

- HE.Add(c_1, c_2): It takes the ciphertexts c_1 and c_2 of m_1 and m_2 as input and outputs the ciphertext c_{add} of $m_1 + m_2$.
- HE.Mul(c_1, c_2): It takes the ciphertexts c_1 and c_2 of m_1 and m_2 as input and outputs the ciphertext c_{mul} of $m_1 * m_2$.
- HE.Rot(c, n): It takes the ciphertexts c and a number of positions n as input and outputs the rotated ciphertext c_r.

3.2 Message locked encryption

MLE generates encryption keys based on the content of the plaintext, ensuring that identical plaintexts produce identical ciphertexts. This property enables encrypted deduplication. MLE typically includes the following algorithms:

- MLE.KeyGen(m): It takes a message m as input and outputs a key k.
- MLE.Enc(m, k): It takes a message m and a key k as input and outputs a ciphertext c.
- MLE.Dec(c, k): It takes a ciphertext c and a key k as input and outputs a message m.

3.3 Fasta

From the proposal of HHE [20], many scholars have evaluated existing symmetric encryption schemes. Although certain progress has been made, existing symmetric encryption schemes are still not suitable for HE. Therefore, many scholars have begun to propose symmetric encryption schemes suitable for HE. Fasta [3] is one such scheme. Based on HElib implementation, Fasta can be seen as a variant of Rasta [4], but with improvements to enable efficient implementation on BGV. The Fasta stream encryption mainly includes the following algorithms:

- Fasta.KeyGen(λ): It takes a security parameter λ as input and outputs a key K.
- Fasta.KeySGen$\varepsilon(K)$: It takes a key K as input and outputs a keystream k_1, k_2, k_3, k_4, k_5.
- Fasta.Enc$(k_1, k_2, k_3, k_4, k_5, m)$: It takes a keystream k_1, k_2, k_3, k_4, k_5 and a message m as input and outputs a ciphertext c of m.
- Fasta.Dec$(k_1, k_2, k_3, k_4, k_5, c)$: It takes a keystream k_1, k_2, k_3, k_4, k_5 and a ciphertext c as input and outputs a message m.

3.4 K-modes

Despite being a simple and efficient clustering algorithm, the conventional K-means algorithm and the improved K-means++ [1] algorithm are only suitable for datasets with continuous attributes. For datasets with discrete attributes, calculating the mean of clusters and the Euclidean distance between points becomes inappropriate. To overcome the drawbacks of partition-based clustering

https://github.com/homenc/HElib

algorithms like K-means, Huang *et al.* [13] proposed the K-modes clustering algorithm.

As an extension of K-means, K-modes uses Hamming distance and is suitable for datasets with discrete attributes. The concept of K-modes is relatively simple, and its time complexity is lower than that of K-means and K-medoids [15]. The K-modes algorithm mainly includes the following steps:

- $\mathrm{Dis}(d_i, d_j)$: This function is used to calculate the Hamming distance. It takes two data points d_i and d_j from the dataset as input and outputs the Hamming distance between the two data points.
- $\mathrm{initModes}(D, k)$: This function is used to initialize the cluster centers. It takes a dataset D and the initial number of desired cluster centers k as input and outputs the initially selected cluster centers C.
- $\mathrm{assignPToM}(D, C)$: This function is used to assign data points to the cluster centers. It takes a dataset D and the cluster centers C as input, and outputs the set A where each data point is assigned to the nearest cluster center based on distance.
- $\mathrm{update}(D, C, A)$: This function is used to update the cluster centers. It takes a dataset D, cluster centers C, and the set of assignments A as input, and outputs the new cluster centers C_{new} and the updated set of assignments A_{new}.

4 Our scheme

4.1 Main idea

There are several issues that need to be resolved in the deduplication and approximate analytics for encrypted IoT data.

The first issue is how to encrypt and deduplicate similar but distinct IoT data to save storage costs and protect privacy. Given the large amount of redundancy in IoT data, directly storing this data can result in significant storage costs. Therefore, it is necessary to process the IoT data to remove redundant content. However, if the deduplicated data is stored in plaintext, it increases the risk of data breaches. To address this, we apply MLE to perform encrypted deduplication on similar but distinct IoT data. We first use GD to separate similar but distinct IoT data into bases and deviations. MLE and Fasta are combined to encrypt and deduplicate the bases, while Fasta is used to encrypt the deviations. The encrypted bases are stored in the fog nodes, and the encrypted deviations are stored in the cloud server. This is because the bases tend to have a higher repetition rate and occupy less space after deduplication, yet they contain more information. In contrast, deviations contain less information. By storing the ciphertexts of the bases in the fog nodes, it facilitates subsequent approximate analytics by the fog nodes using the ciphertexts of the bases.

The second issue is how to perform approximate analytics on encrypted IoT data. Conventional symmetric encryption, while efficient, does not permit computations on ciphertext. To address this, we employ HE to secure the data. HE allows for addition and multiplication operations performed on ciphertexts to yield the same result as if these operations were performed on the plaintext and then encrypted. Consequently, IoT data encrypted with HE can undergo approximate analytics.

The third issue addresses the challenge of alleviating computational strain on devices and reducing bandwidth overhead. HE, compared to conventional encryption methods, requires greater computational resources such as CPU and memory. Additionally, data processed through HE undergoes significant volume expansion, increasing the computational burden on devices. Moreover, due to the ciphertext expansion inherent in HE, transmitting data to fog nodes consumes substantial bandwidth. To mitigate these issues, we employ HHE technique to transform symmetric encryption into homomorphic encryption. Devices initially encrypt messages using a designated symmetric encryption method and encrypt the MLE keys using the homomorphic public keys of the Crypto-service Provider (CSP). The devices then send encrypted data and encrypted keys to the fog nodes. Since only the keys are homomorphically encrypted, this results in minimal expansion of the ciphertext, thus reducing communication overhead between devices and fog nodes. Furthermore, devices primarily perform symmetric encryption, which lessens their computational load. The fog nodes decrypt the symmetrically encrypted data on a homomorphic basis, converting symmetric ciphertext into homomorphic ciphertext for further analytics. Fasta, a homomorphic-friendly symmetric encryption scheme, is selected for device encryption.

4.2 Architecture

This section describes the design of the framework for our scheme.

As illustrated in Fig. 3, our architecture comprises four components: IoT device, fog node, cloud server, and CSP.

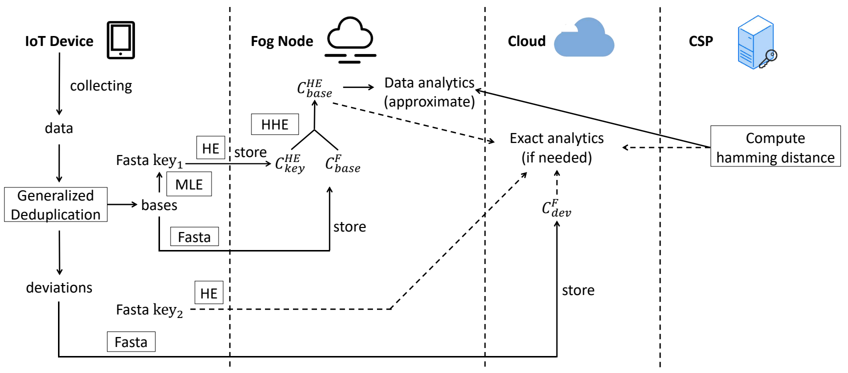


Fig. 3: The architecture of our scheme.

- *IoT device*: IoT device is responsible for collecting and initially processing data. After collecting data, the IoT device first applys GD to process the data, resulting in bases and deviations. The IoT device uses MLE.KeyGen() to generate keys to encrypt the bases using Fasta.Enc() and generates keys to encrypt the deviations. The encrypted data is then sent to fog nodes and cloud server for storage, thereby saving local storage overhead. Additionally, the device also uses HE.Enc() to encrypt the keys and uploads the homomorphic ciphertext of the keys to the fog nodes.
- *Fog node*: Upon receiving the data from the devices, the fog nodes store the homomorphically encrypted keys alongside the symmetrically encrypted data. When analytics is required, these are retrieved and, using HHE techniques, the symmetrically encrypted data is converted into homomorphically encrypted data for subsequent analytics.
- *Cloud server*: The cloud server stores the deviations encrypted symmetrically by the devices. If exact analytics is required, the cloud server can receive the homomorphic ciphertext of the bases from the fog node, the homomorphic ciphertext of the encryption key for the deviations from the IoT device, and process them to perform exact analytics.
- *CSP*: The CSP manages the key pair for the HE. Additionally, during approximate analytics at the fog node, the fog node sends the homomorphically encrypted XOR result to the CSP. The CSP decrypts it, calculates the Hamming distance, and returns the result to the fog node.

4.3 Threat model

We assume that the fog node, cloud server, and CSP are all honest-but-curious parties. In our scheme, since the homomorphic encryption public key pk used by the devices is provided by CSP, the leakage of CSP private key sk would enable attackers to access all data information. Therefore, it is crucial for CSP to securely store the homomorphic encryption private key sk. Additionally, we consider the following two types of adversaries:

- Honest but curious fog nodes and cloud servers execute our proposed scheme faithfully, yet they attempt to extract as much information as possible based on the stored data. Both fog nodes and cloud servers can access all data stored within them and may try to decrypt the data to recover the original information.
- Devices honestly execute our scheme for data upload. Since our scheme performs deduplication on the server side, devices cannot carry out side-channel attacks [12] or duplicate faking attacks.
- Honest but curious CSP execute our proposed scheme faithfully, yet it tries to infer the original data from the decrypted XOR result.

4.4 Construction

In this section, we present the construction of the algorithms and protocols within our scheme.

Table 1: Major notations used in this paper.

Notation	Description
k	MLE key
pk, sk	Key pair of CSP
π_I	GD permutation function
ϕ_k	GD transformation function
b, d	Bases and deviations
C_k^{HE}	Homomorphic ciphertext of MLE key
C_b^{Fa}, C_d^{Fa}	Fasta ciphertext of bases and deviations
$C_b^{Fa,HE}$	Homomorphic ciphertext of Fasta encrypted bases
$Eval$	Ciphertext transformation function

Key generation(KG). Our scheme follows the key generation algorithms for HE and MLE. During system setup, CSP generate homomorphic keys pk and sk using HE.KeyGen(), and securely store the private key sk. When devices join the system, they interact with CSP to obtain the public key pk.

Preprocessing generalized deduplication(Pre-GD). To employ the GD suitable for the data collected by the devices, we initially select a certain number of samples from the device-collected data for preprocessing training. We preprocess the training set using the algorithm described in the [28] to determine the appropriate sample concatenation length parameter c. Subsequently, based on the formula $S_k = Kk + N(\lceil \log_2 K \rceil + (n-k))$ [28], we derive the most suitable permutation function π_I and transformation function ϕ_k using the training set, and store them in the device for subsequent processing.

Generalized deduplication(GD). Initially, the device retrieves the previously stored parameters c, permutation function π_I, and transformation function ϕ_k, and then concatenates and inputs the collected data. Subsequently, the data is transformed into bases and deviations. Since the bases contain the majority of redundant content, it is deduplicated, meaning that only one copy of the same data is retained. As the deviations contain less redundancy and are smaller in volume, they are not deduplicated. Thus all deviations are preserved. After processing, the device separates the data into the bases b and deviations d.

Data upload. As illustrated in Fig. 4, after the data has been processed using the GD, the device uses MLE.KeyGen(m) to generate k. Then, the device uses k and the symmetric encryption algorithm Fasta to encrypt the bases b and deviations d, producing C_b^{Fa} and C_d^{Fa}. Subsequently, the device uses the fog node's public key pk to perform HE.Enc() on the key k, generating C_k^{HE}. Once the encryption is completed, the device sends C_d^{Fa} to the cloud server for storage, and C_b^{Fa} along with C_k^{HE} to the fog node for storage.

Data analytics. As illustrated in Fig. 5, when approximate analytics is required, the fog node retrieves the stored C_b^{Fa} and C_k^{HE}. Then, C_b^{Fa} is homomorphically encrypted, producing $C_b^{Fa,HE}$. Following this, based on the principle

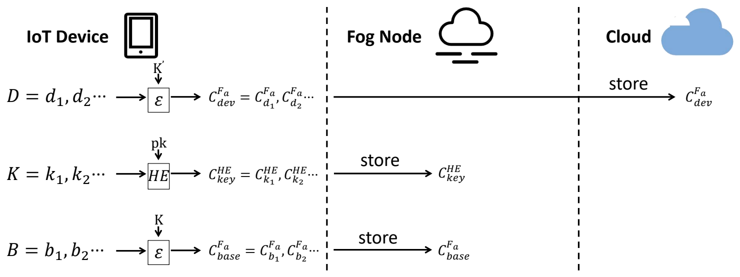


Fig. 4: Data upload. ε represents the stream key generation and encryption processes of the Fasta stream cipher.

of converting symmetric encryption into HE within HHE, Eval$(C_b^{Fa,HE}, C_k^{HE})$ is used to transform the symmetrically encrypted $C_b^{Fa,HE}$ into homomorphically encrypted C_b^{HE}. Subsequently, K-modes clustering approximate analytics is conducted. Initially, a random sample is selected from the homomorphically encrypted dataset to serve as the initial clustering center. Subsequently, the shortest hamming distance between other data points and the clustering center is calculated, followed by the selection of the next clustering center. This process is iterated until k clustering centers have been selected or the specified number of iterations has been reached. When calculating the Hamming distance, the fog node computes the XOR value of the two homomorphically encrypted points and sends it to the CSP. The CSP decrypts the XOR value, calculates the Hamming distance, and returns the result to the fog node.

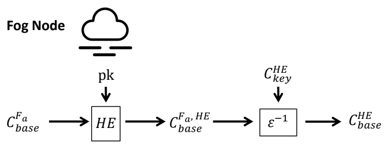


Fig. 5: Ciphertext transformation on the fog node.

4.5 Discussion

In this section, we illustrate that our scheme can also support exact analytics. Since our scheme utilizes HE based on bits, it allows for not only approximate analytics of the data but also exact analytics of the data. When conducting exact analytics, if there has been an approximate analytics process previously, the fog node can directly send the homomorphically encrypted base to the cloud server. Although this increases communication overhead, it eliminates the need for the cloud server to perform ciphertext transformation. If the fog node has not transformed the data, it must send both the symmetrically encrypted base and the homomorphically encrypted MLE keys to the cloud server for ciphertext transformation. The cloud server also requests the device to upload the homomorphic ciphertext of the encryption keys for the deviations and performs ciphertext transformation on the deviations. Since our encryption is performed bit by bit, we can recover the original dataset based on the GD recovery process without performing homomorphic decryption, thus protecting data privacy.

However, as all data is recovered, the volume of the corresponding homomorphic ciphertext will be large, resulting in significantly increased computation overhead, storage, and analytics time costs, substantially affecting system performance. Therefore, our scheme primarily focuses on approximate analytics, and we plan to design and improve the exact analytics system in future work.

5 Security analytics

In this section, we provide a security analysis of our scheme based on our threat model (see Section 4.3).

The fog nodes receive encrypted bases and keys from IoT devices. The bases are encrypted using Fasta, while the keys are encrypted using HE and securely stored. The fog nodes perform ciphertext transformation entirely in the encrypted state, ensuring no information is exposed. Similarly, the cloud servers receive encrypted deviations from the IoT devices. These deviations are encrypted using Fasta. Since both the Fasta and HE are secure, neither the fog nodes nor the cloud servers can recover the data.

Furthermore, since the data used in approximate analytics is based on homomorphically encrypted data, no data information is retained in approximate analytics.

Honest but curious CSP can only learn the XOR value of two data computations. With just the XOR value, the CSP cannot recover any information. Since our scheme uses base data with a length of 329 bits, the probability of directly guessing the correct two data values is negligible. Moreover, the CSP will only obtain an XOR result. Even if the CSP selects the corresponding data, it cannot determine whether the selection is correct.

6 Evaluation

We implemented our prototype based on the HElib library. The prototype comprises IoT device, fog node, cloud server, and CSP. The HHE used in the prototype combines the BGV algorithm and the Fasta stream encryption algorithm. The security level for HE is set to 128 bits, and Fasta is configured according to the security parameters provided in the [3]. Our experiments are conducted on an Ubuntu 20.04.6 virtual machine created on a laptop equipped with an AMD Ryzen 5 5600H processor (3.30 GHz), 16 GB (15.9 GB usable) memory, based on x64 architecture 64-bit operating system. The virtual machine has 20 GB of memory, 8 GB of RAM, and an 8-core processor. The evaluation results presented in this paper are the averages of over 10 runs.

We used a real-world dataset [17], ECA&D , to evaluate the performance of our scheme. ECA&D collects data on weather and climate extremes across Europe. We selected the wind speed meteorological data for testing. Each piece of data is 15B in size before processing. For our evaluation, we selected data from Iceland and Italy from this dataset.

Performance of GD for compressing IoT data. We apply GD to similar but distinct weather monitoring IoT data from Iceland and Italy, dividing the data into bases and deviations. The Iceland data we used is 3030 KB; after deduplication, the base data is 45 KB and the deviation data is 569 KB, achieving a compression rate of nearly 80%. The Italy data we used is 2356 KB; after deduplication, the base data is 24 KB and the deviation data is 457 KB, also achieving a compression rate of nearly 80%. This demonstrates that GD can significantly reduce the storage overhead for similar but distinct IoT data.

Additionally, we tested the compression rates achieved by applying GD to datasets from multiple countries within the entire dataset. The results are shown in Table 2.

Table 2: The compression rates after using GD on datasets from different countries. The formula for compression rates is $\left(1 - \frac{\text{Compressed Data Size}}{\text{Original Data Size}}\right) \times 100\%$

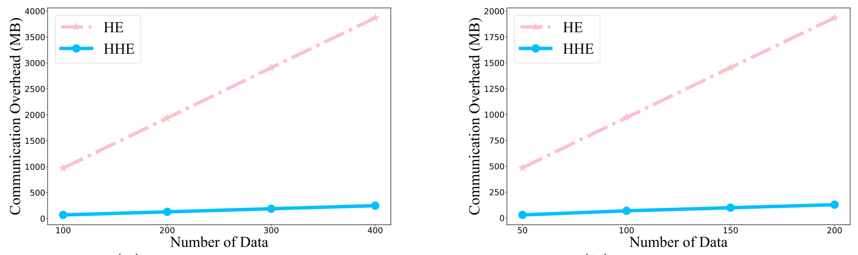
Country	Albania	Iceland	Poland	Finland	Italy
Percentage	76.63%	77.94%	80.30%	79.46%	81.00%

Country	Montenegro	Norway	Portugal	Hungary	UK
Percentage	76.86%	83.91%	79.80%	82.44%	80.50%

Performance of using HHE by fog nodes and IoT devices. We evaluate the communication overhead and encryption time of IoT devices and fog nodes using HHE and compare it with the scenario where only HE is used. First, we assess the communication overhead for both HHE and solely HE. As shown in the Fig. 6(a), we use the Iceland dataset for evaluation. When directly encrypting messages using HE and uploading them, each message reaches up to 9.7 MB (9,715,003 bytes) after encryption. If the entire dataset is uploaded, the communication overhead will be as high as 3.84 GB. In contrast, with our scheme, we encrypt the base key, which results in an encrypted base key size of 9.7 MB (9,715,562 bytes), and each base key symmetrically encrypts 16 messages, totaling 1645 bits. Uploading the entire dataset in our scheme results in a communication overhead of 250.17 MB, which reduces the communication overhead by approximately 93.6%. Similarly, as shown in the Fig. 6(b), for the Italian dataset, the communication overhead when using only HE is around 2.19

The European Climate Assessment & Dataset project, https://www.ecad.eu.

GB, whereas using our scheme, the communication overhead is approximately 148.51 MB, reducing it by about 93.5%. As the amount of data transmitted increases, the communication overhead of using only HE becomes increasingly unacceptable. Our scheme significantly reduces the communication overhead between devices and fog nodes. This reduction is achieved because our scheme employs HHE, allowing us to encrypt only the base keys homomorphically, thus effectively reducing ciphertext expansion. In contrast, directly using HE results in significant ciphertext expansion.

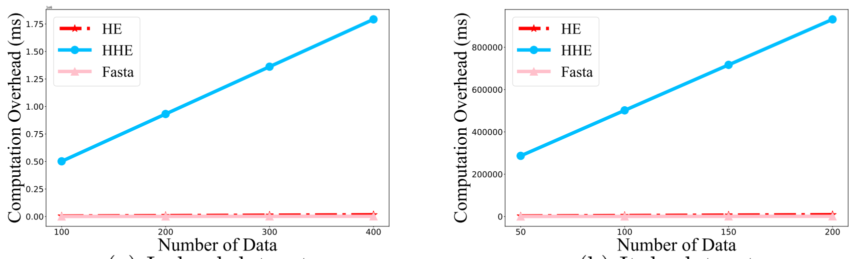


(a) Iceland datase (b) Italy dataset

Fig. 6: Communication overhead between IoT devices and fog nodes using the Iceland dataset and the Italy dataset.

Next, we evaluate the encryption time of IoT devices and fog nodes using HHE compared to using only HE. We encrypt 16 pieces of data as a group. As shown in the Fig. 7, If the IoT device directly encrypts a group of data using HE, it takes 640ms. However, if the IoT device uses our scheme, it performs Fasta encryption, which takes only 76.3ms to encrypt a group of data. Our scheme significantly reduces the encryption time for IoT devices. When the fog node needs to perform approximate analytics, it needs to homomorphically generate a keystream and then perform symmetric decryption. This process takes 69,284ms to generate the keystream, and generating the keystream and symmetrically decrypting a group of data takes 71,683ms in total (in our tests of HHE, we did not enable parallel processing; theoretically, the time for homomorphic key stream generation and symmetric decryption can be reduced to approximately 50ms). Our scheme notably reduces the encryption time for IoT devices.

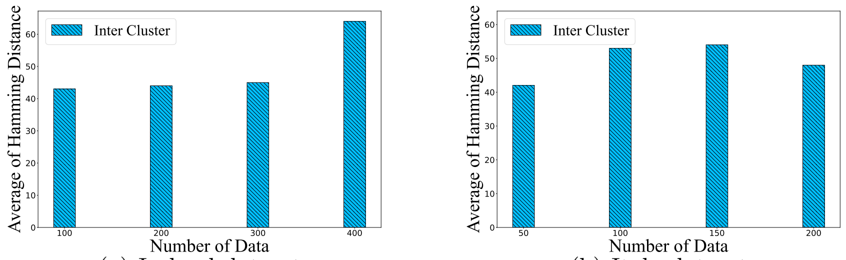
Our scheme significantly reduces the communication overhead between IoT devices and fog nodes, as well as the encryption time for IoT devices. Although our scheme introduces additional time consumption when approximate analytics is required, this additional time is relatively small compared to the overall time required for subsequent approximate analytics. Furthermore, our scheme alleviates the burden on IoT devices, which typically have weaker performance, thereby significantly reducing the overall time and performance loss. Therefore, despite the additional time introduced for the fog nodes, this overhead is acceptable.



(a) Iceland dataset (b) Italy dataset

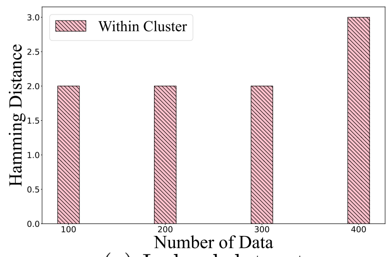
Fig. 7: Comparison of encryption times using the Iceland dataset and the Italy dataset.

Performance of approximate analytics. We used three metrics to evaluate the performance of K-modes. First, the sum of Hamming distances within each cluster and the Hamming distances between cluster centers are used to assess the clustering quality. For a given dataset, smaller within-cluster Hamming distances and larger Hamming distances between cluster centers indicate higher quality (tighter) clusters. Secondly, to compare the quality of cluster centers obtained from compressed and uncompressed data, we used the elbow method for both datasets. The smaller the difference in "elbow" values between the two datasets, the better the clustering quality will be.

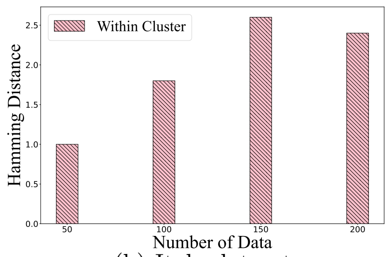


(a) Iceland dataset (b) Italy dataset

Fig. 8: The Hamming distance between K-modes cluster centers using the Iceland dataset and the Italy dataset.



(a) Iceland dataset

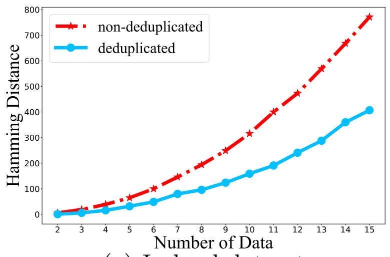


(b) Italy dataset

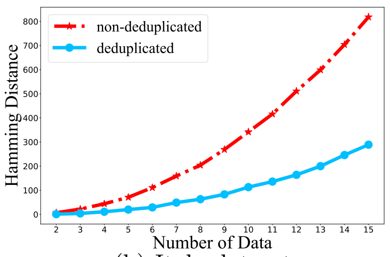
Fig. 9: The Hamming distance between data within a cluster and the cluster center using the Iceland dataset and the Italy dataset.

After plotting the elbow curve, we selected the optimal number of cluster centers to be 7. As shown in Fig. 8 and Fig. 9, the average Hamming distance within clusters obtained through K-modes approximate analytics is significantly smaller than the average Hamming distance between cluster centers. Therefore, the effectiveness of our approximate analytics is well demonstrated.

Through Fig. 10, it can be observed that the elbow points obtained using approximate analytics and those obtained using exact analytics are almost identical. Additionally, the cluster centers obtained are also nearly the same. Therefore, the data obtained through approximate analytics has significant practical value.



(a) Iceland dataset



(b) Italy dataset

Fig. 10: The elbow curve for approximate analytics and exact analytics using the Iceland dataset and the Italy dataset.

Performance of computation time for approximate analytics of homomorphically encrypted data. We selected data from the Iceland dataset for evaluation. We evaluated the time required for approximate analytics using different amounts of homomorphically encrypted data, as shown in Fig. 11. Due to the substantial time consumption of homomorphic operations, the runtime of our scheme is relatively favorable.

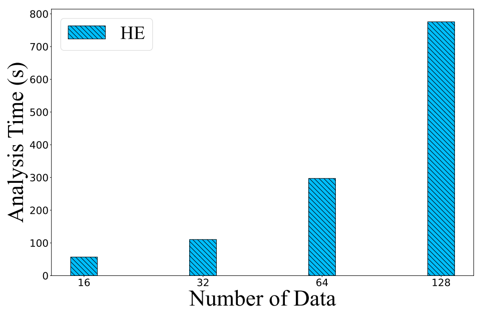


Fig. 11: Computation time for approximate analytics.

7 Conclusion

In this paper, we propose a deduplication and approximate analytics scheme for encrypted IoT data in fog-assisted cloud storage. Our approach combines GD and MLE to achieve encrypted deduplication of similar but distinct IoT data, thereby reducing storage overhead. Grounded on HE, our approach enables approximate analytics without revealing data content, further safeguarding data privacy. Additionally, we leverage HHE technology, significantly alleviating the computational burden on IoT devices while reducing communication overhead between IoT devices and fog nodes. By offloading certain computational tasks to the fog nodes for processing, we effectively lighten the load on devices, enhancing the overall efficiency of the system. Moreover, we conduct a thorough security analytics of the solution, ensuring confidentiality for similar but distinct IoT data. Finally, through performance evaluation using real-world IoT datasets, our results demonstrate the solution's capability to effectively reduce storage overhead and significantly lower communication costs.

Our solution is also capable of performing exact analytics, albeit at an unacceptable computational and storage cost. Our future work involves further enhancing the computational performance of the solution and exploring methods to implement efficient exact analytics.

References

1. Arthur, D., Vassilvitskii, S.: K-means++: The advantages of careful seeding. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1027–1035 (2007)
2. Bellare, M., Keelveedhi, S.: Interactive message-locked encryption and secure deduplication. In: Katz, J. (ed.) Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9020, pp. 516–538. Springer (2015)
3. Cid, C., Indrøy, J.P., Raddum, H.: FASTA - A stream cipher for fast FHE evaluation. In: Galbraith, S.D. (ed.) Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13161, pp. 451–483. Springer (2022)
4. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: A cipher with low anddepth and few ands per bit. IACR Cryptol. ePrint Arch. p. 181 (2018)
5. Douceur, J.R., Adya, A., Bolosky, W.J., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, July 2-5, 2002. pp. 617–624. IEEE Computer Society (2002)
6. Facebook: Zstandard–real-time data compression algorithm (2021), [Online]. Available: https://facebook.github.io/zstd/ (Accessed: Dec. 22, 2021)
7. Frimpong, E., Nguyen, K., Budzys, M., Khan, T., Michalas, A.: Guardml: Efficient privacy-preserving machine learning services through hybrid homomorphic encryption. CoRR **abs/2401.14840** (2024)
8. Gao, Y., Chen, L., Han, J., Yu, S., Fang, H.: Similarity-based secure deduplication for iiot cloud management system. IEEE Transactions on Dependable and Secure Computing (2023)
9. Gao, Y., Chen, L., Han, J., Yu, S., Fang, H.: Similarity-based secure deduplication for iiot cloud management system. IEEE Trans. Dependable Secur. Comput. **21**(4), 2242–2256 (2024)
10. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC. pp. 169–178. ACM (2009)
11. Ha, G., Jia, C., Chen, Y., Chen, H., Li, M.: A secure client-side deduplication scheme based on updatable server-aided encryption. IEEE Trans. Cloud Comput. **11**(4), 3672–3684 (2023)
12. Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services: Deduplication in cloud storage. IEEE Secur. Priv. **8**(6), 40–47 (2010)
13. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**(3), 283–304 (1998)
14. Hurst, A., Lucani, D.E., Assent, I., Zhang, Q.: GLEAN: generalized-deduplication-enabled approximate edge analytics. IEEE Internet Things J. **10**(5), 4006–4020 (2023)
15. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley (1990)
16. Keelveedhi, S., Bellare, M., Ristenpart, T.: Dupless: Server-aided encryption for deduplicated storage. In: King, S.T. (ed.) Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013. pp. 179–194. USENIX Association (2013)

17. Klein Tank, A.M.G., Coauthors: Daily dataset of 20th-century surface air temperature and precipitation series for the european climate assessment. International Journal of Climatology **22**, 1441–1453 (2002)
18. Li, J., Yang, Z., Ren, Y., Lee, P.P.C., Zhang, X.: Balancing storage efficiency and data confidentiality with tunable encrypted deduplication. In: Bilas, A., Magoutis, K., Markatos, E.P., Kostic, D., Seltzer, M.I. (eds.) EuroSys '20: Fifteenth EuroSys Conference 2020, Heraklion, Greece, April 27-30, 2020. pp. 22:1–22:15. ACM (2020)
19. Miao, M., Wang, J., Li, H., Chen, X.: Secure multi-server-aided data deduplication in cloud computing. Pervasive Mob. Comput. **24**, 129–137 (2015)
20. Naehrig, M., Lauter, K.E., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Cachin, C., Ristenpart, T. (eds.) Proceedings of the 3rd ACM Cloud Computing Security Workshop, CCSW 2011, Chicago, IL, USA, October 21, 2011. pp. 113–124. ACM (2011)
21. Nord, J., Koohang, A., Paliszkiewicz, J.: The internet of things: Review and theoretical framework. Expert Systems with Applications **133**, 97–108 (2019)
22. Qin, C., Li, J., Lee, P.P.C.: The design and implementation of a rekeying-aware encrypted deduplication storage system. ACM Trans. Storage **13**(1), 9:1–9:30 (2017)
23. Sampson, A., Dietl, W., Fortuna, E., Gnanapragasam, D., Ceze, L., Grossman, D.: Enerj: approximate data types for safe and general low-power computation. In: Hall, M.W., Padua, D.A. (eds.) Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2011, San Jose, CA, USA, June 4-8, 2011. pp. 164–174. ACM (2011)
24. Services, H.B.R.A.: Accelerating the internet of things timeline (2019)
25. Seward, J.: bzip2 (2019), [Online]. Available: https://sourceware.org/bzip2/
26. Shin, Y., Koo, D., Yun, J., Hur, J.: Decentralized server-aided encryption for secure deduplication in cloud storage. IEEE Trans. Serv. Comput. **13**(6), 1021–1033 (2020)
27. Song, M., Hua, Z., Zheng, Y., Xiang, T., Jia, X.: Fcdedup: A twolevel deduplication system for encrypted data in fog computing. IEEE Transactions on Parallel and Distributed Systems (2023)
28. Vestergaard, R., Lucani, D.E., Zhang, Q.: A randomly accessible lossless compression scheme for time-series data. In: 39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020. pp. 2145–2154. IEEE (2020)
29. Vestergaard, R., Zhang, Q., Lucani, D.E.: Generalized deduplication: Bounds, convergence, and asymptotic properties. In: 2019 IEEE Global Communications Conference, GLOBECOM 2019, Waikoloa, HI, USA, December 9-13, 2019. pp. 1–6. IEEE (2019)
30. Wang, C., Zhou, T., Shen, J., Wang, W., Zhou, X.: Searchable and secure edge precache scheme for intelligent 6g wireless systems. Future Generation Computer Systems **140**, 129–137 (2023)
31. Weber, R.H.: Internet of things: Privacy issues revisited. Computer Law & Security Review **31**(5), 618–627 (2015)
32. Wood, A., Najarian, K., Kahrobaei, D.: Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Comput. Surv. **53**(4), 70:1–70:35 (2021)
33. Zhang, C., Miao, Y., Xie, Q., Guo, Y., Du, H., Jia, X.: Privacy-preserving deduplication of sensor compressed data in distributed fog computing. IEEE Transactions on Parallel and Distributed Systems **33**(12), 4176–4191 (2022)