Communication Volume Reduction of Homomorphic Ciphertext with Compressed Sensing

Koyuki Izumi
Ochanomizu University
Tokyo, Japan
koyuki@ogl.is.ocha.ac.jp

Marin Matsumoto
Ochanomizu University
Tokyo, Japan
marin@ogl.is.ocha.ac.jp

Masato Oguchi
Ochanomizu University
Tokyo, Japan
oguchi@is.ocha.ac.jp

***Abstract*—Homomorphic encryption (HE) enables the addition and multiplication of ciphertexts, which is not possible with most other encryption methods and is a technique that can be analyzed and utilized while maintaining data secrecy. Examples of HE applications include encrypted databases and regression analyses. The data encrypted by the client can be aggregated/analyzed by the database server, and the data analyst can only obtain the aggregated/analyzed results. However, in such applications, the size of the homomorphic ciphertext is larger than that of plaintext, increasing the communication between the client and the database server. In this study, we propose a method for reducing the communication volume by compressed sensing using a dictionary. In the proposed method, the client encrypts dimensionally compressed records, and the database server reconstructs the original records using the dictionary while still encrypting, thus reducing the size of the communication between the client and the database server for data aggregation/analysis while maintaining the accuracy of the reconstruction. The evaluation results demonstrate that when the compression ratio is set to 50%, the proposed method can reduce the communication volume by 50% compared to the case where the data are transmitted without compression. We demonstrate the utility of compressed/reconstructed data based on the accuracy of binary classification using logistic regression.**

***Index Terms*—Homomorphic Encryption, Compressed Sensing**

I. Introduction

Since the data collected from Internet of Things (IoT) devices such as smartphones may contain personal information, preventing data leakage in cloud servers is necessary, and encryption is essential. In most other encryption methods, the addition and multiplication of ciphertexts are not possible, but homomorphic encryption (HE) is a technique that makes these operations possible. Therefore, HE is helpful for data utilization while maintaining data secret. Examples of HE applications include encrypted databases and regression analyses. However, owing to the nature of HE, where the size of ciphertext is larger than that of plaintext, the amount of communication between the client and database server increases. In this study, we propose reducing this communication volume using compressed sensing, as shown in Figure 1. In the proposed method, multidimensional plaintexts are compressed to low dimensions and then encrypted with HE to reduce the amount of communication between the client and server. The server then reconstructs the multidimensional data from the low-dimensional data under encryption. Compressed sensing is a technique that uses sparsity to recover the original data from a small number of samples. It compresses the data into low-dimensional data $\boldsymbol{y}$ by multiplying by a compression matrix $\mathbf{A}$ and estimates the original multidimensional data $\boldsymbol{x}_0$ using the obtained low-dimensional data and compression matrix.

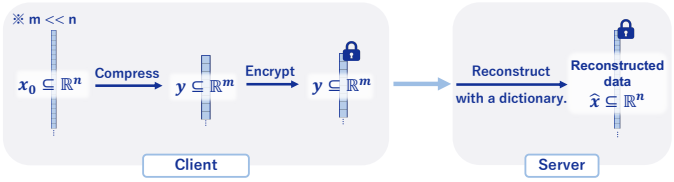


Fig. 1: Our Proposal: On the client side, data is compressed first and then encrypted. Since the compressed data is sent from the client to the server, communication volume can be reduced. The server reconstructs the data using the dictionary when it is needed.

Our challenges. Two challenges exist to adapting compressed sensing to HE. The first challenge is the accuracy of reconstruction. Compressed sensing assumes sparsity in the original data. If the data are not sparse, reconstruction accuracy is degraded. Therefore, we employed a dictionary created using K-singular value decomposition (K-SVD). The dictionary is learned beforehand using data from the same domain as the client data. On the server side, the dictionary is used for reconstruction through compressed sensing with encryption. The second challenge involves the reconstruction algorithm for HE. As HE is not suitable for comparison operations, the reconstruction algorithm must be carefully considered. Moreover, the higher the number of multiplications included in the reconstruction algorithm, the longer the execution time. To address these limitations, the server estimates $\boldsymbol{x_0}$ by L_2 norm minimization, using the generalized inverse of the proposed method.

Contributions. The contributions of this study are as follows:

- The proposed method achieved a 50% reduction in HE ciphertext size by compressed sensing.
- We analyzed the number of homomorphic operations of the reconstruction algorithm and showed that splitting the compressed vector into smaller parts is more efficient.
- By applying dictionary learning, the proposed method

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

maintained its recovery accuracy even for non-sparse multidimensional data. Dictionary learning reduces the accuracy of binary classification through logistic regression due to compression by only approximately 0.3 – 1.6%.

II. Related work

Most studies on the optimization of HE have mainly focused on the acceleration of homomorphic operations [1]–[3], whereas few studies [4], [5] have addressed the reduction in the size of ciphertext. Matsumoto et al. [4] proposed a combination of lightweight additive homomorphic encryption (AHE) and leveled homomorphic encryption (LHE) to reduce the load of LHE in IoT devices. In this method, plaintext is encrypted with AHE on the IoT device side. The cloud server switches from an AHE ciphertext to an LHE ciphertext. They showed that the proposed method reduces the size of ciphertext by approximately 73 – 233 times compared with existing methods. However, this method increases the number of keys that must be managed. Our proposed method can reduce ciphertext size without increasing the number of keys to be managed.

HE is a promising candidate for secure data outsourcing; however, supporting real-world machine learning (ML) tasks is challenging. Kim et al. [6] performed a logistic regression analysis based on HE by approximating a sigmoid function. Logistic regression uses a sigmoid function $\frac{1}{1+e^{-x}}$; however, because calculating the sigmoid function directly in an HE state is difficult, the sigmoid function was approximated using the least squares method. Experimental results on the dataset demonstrate that the best case is when the least squares approximation of a seventh-degree polynomial is used and that the results obtained from logistic regression analysis on the plaintext and logistic regression analysis on an HE is almost equivalent. These results indicate that the encrypted logistic regression can learn with high accuracy while preserving privacy.

III. Preliminaries

A. Homomorphic encryption

Homomorphic encryption is a technique that allows computations on ciphertexts and generates encrypted results that match those of plaintext computations. This technology has significant potential for real-world applications because it allows us to securely outsource expensive computations on an untrusted server.

HE can be classified into four generations. In particular, the 4th generation CKKS scheme [7] can calculate integers as well as real and complex numbers. The CKKS scheme has garnered considerable attention in recent years because it can be applied to ML [8].

B. Compressed sensing

Compressed sensing [9] is a framework for reconstructing high-dimensional signals with sparsity (the property of having many zero components) based on a few observations. Suppose that an unknown n-dimensional vector $\boldsymbol{x_0}$ is linearly transformed into $\boldsymbol{y} = \mathbf{A}\boldsymbol{x_0}$ using $\mathbf{A} \in \mathbb{R}^{m\times n}$, where $m < n$.

In general, the number of unknowns is greater than that of equations; therefore, it is indefinite, and the solution cannot be uniquely determined. However, we can uniquely estimate the solution if many elements of $\boldsymbol{x_0}$ are zero, that is, a sparse vector.

Reconstruction algorithm. Several reconstruction algorithms [10]–[12] have been used in compressed sensing. Here, we introduce a representative algorithm, orthogonal matching pursuit (OMP) [12]. OMP is a greedy algorithm that approximates $\boldsymbol{y}$ using only the components of column vector $\boldsymbol{a}_i$. It reconstructs the data by adding the indices of the column vectors of $\mathbf{A}$ to index set T and in the order of the best approximation of the residuals. Generally, $\hat{\boldsymbol{x}}$, the solution to $\mathrm{argmin}||\boldsymbol{y} - \mathbf{A_T}\boldsymbol{x_T}||_2^2$, can be determined by $\hat{\boldsymbol{x}} = (\mathbf{A_T}^\top\mathbf{A_T})^{-1}\mathbf{A_T}^\top\boldsymbol{y}$.

HE is unsuitable for comparison operations, and if the algorithm is not simple, the number of homomorphic operations increases. Therefore, the reconstruction algorithm must be selected carefully.

Dictionary learning. Dictionary learning is the process of learning optimal sparse transformations from data [13]. Therefore, using a dictionary in compressed sensing makes it possible to reconstruct even non-sparse data. Dictionary learning is used in compressed sensing as follows [14]:

$$\boldsymbol{x_0} = \mathbf{D}\boldsymbol{c} \tag{1}$$

$$\mathbf{A}' = \mathbf{AD} \tag{2}$$

$$\therefore \boldsymbol{y} = \mathbf{A}\boldsymbol{x_0} = \mathbf{AD}\boldsymbol{c} = \mathbf{A}'\boldsymbol{c} \tag{3}$$

When expressed as $\boldsymbol{y} = \mathbf{A}\boldsymbol{x_0}$, dictionary learning-based compressed sensing is expressed as in (1) using dictionary (a set of n-dimensional basis vectors) $\mathbf{D}$ and unknown coupling coefficients $\boldsymbol{c}$. Here, dictionary $\mathbf{D}$ is assumed to have been prelearned using training data such that the coupling coefficients become sparse. If we consider the product of matrix $\mathbf{A}$ and dictionary $\mathbf{D}$ as a new matrix $\mathbf{A}' = \mathbf{AD}$ as in (2), we can estimate coupling coefficients c by reconstructing $\boldsymbol{y} = \mathbf{A}'\boldsymbol{c}$ using compressed sensing. Then, from (1), $\boldsymbol{x_0}$ can be reconstructed by multiplying the estimated coupling coefficient $\boldsymbol{c}$ by dictionary $\mathbf{D}$. The detailed calculations for applying the reconstruction algorithm to (3) are explained in Section IV-C. Several dictionary learning algorithms exist; however, in this study, we used dictionary learning with K-SVD [15], which is one of the most popular. The dictionary learning with K-SVD was implemented based on this reference [16].

IV. System Overview

A. Protocol

The flow of the system using the proposed method is shown in Figure 2 and below.

1) **Dictionary creation:**
   A dictionary is created from data in the same domain as the client's data. In this case, the data used for dictionary

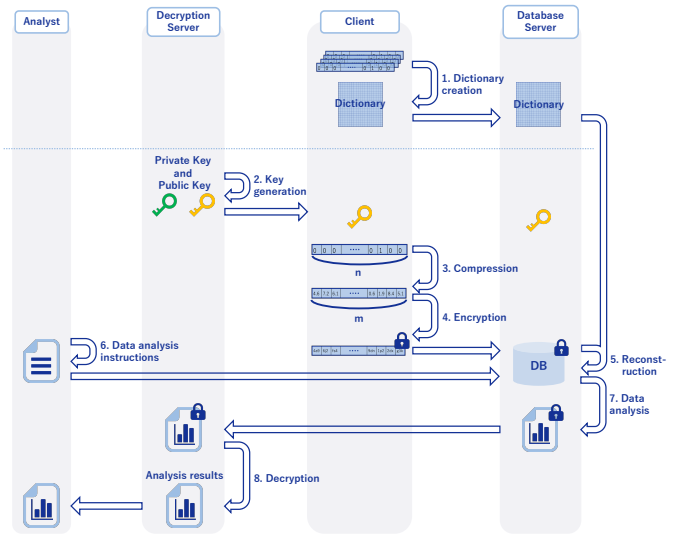


Fig. 2: Processing flow illustration in the proposed method. The system using the proposed method follows the flow of key generation on decryption server, compression and encryption on the client, reconstruction on the database server using the dictionary that is sent in advance, data analysis instructions from the analyst, data analysis on the database server, and decryption on the decryption server.

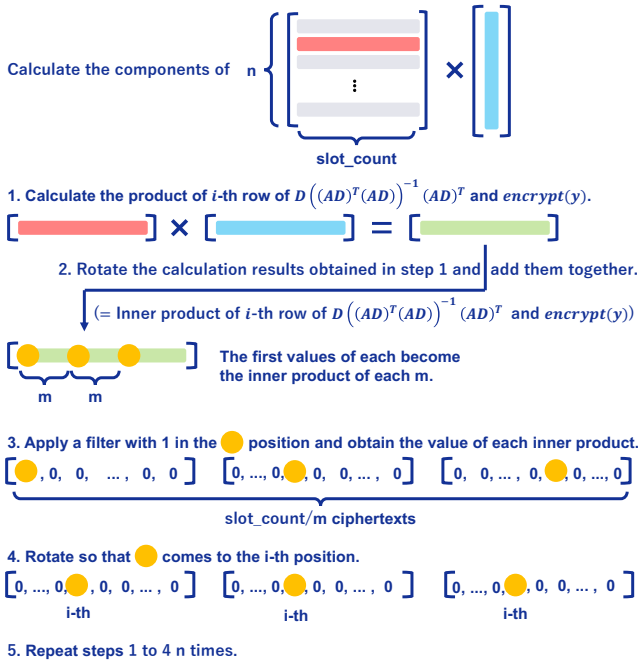


Fig. 3: Reconstruction algorithm under HE. Matrix multiplication was used for the reconstruction. Since matrix multiplication in ciphertexts requires devising calculation, it was calculated using the method shown in this figure.

training is public data that do not overlap with the client's sensitive information. The created dictionary is sent to the database server.

2) **Key generation:**
The decryption server generates the secret and public keys of HE and sends the public key to the client and database server.
3) **Compression:**
The client compresses the plaintext $\boldsymbol{x_0} \subseteq \mathbb{R}^n$. The compressed plaintext is denoted by $\boldsymbol{y} \subseteq \mathbb{R}^m$.
4) **Encryption:**
The client encrypts the slot_count$/m$ compressed plaintexts $\boldsymbol{y}^{\text{slot\_count}/m}$ together and sends it to the database server.
5) **Reconstruction:**
The server reconstructs the ciphertext $encrypt(\hat{\boldsymbol{x}})$ using the dictionary, where $\hat{\boldsymbol{x}} \subseteq \mathbb{R}^n$.
6) **Data analysis instructions:**
The analyst specifies the data analysis.
7) **Data analysis:**
The server performs data analysis on the database server using the reconstructed ciphertext.
8) **Decryption:**
The analyst decrypts the analysis results.

B. Compression and packing

In this study, we use compressed sensing with dictionary learning to apply the proposed method to non-sparse data. When no dictionary is used, an n-dimensional vector is compressed to m dimensions, as in $\boldsymbol{y} = \mathbf{A}\boldsymbol{x_0}$, but when a dictionary is used, we need to consider the formula for compression in (3) again, as the formula is converted from (1), and from (2) to (3). Here, from (1), $\boldsymbol{c} = \mathbf{D^{-1}}\boldsymbol{x_0}$, and if we substitute this into (3), we obtain $\boldsymbol{y} = \mathbf{ADD^{-1}}\boldsymbol{x_0} = \mathbf{A}\boldsymbol{x_0}$. Therefore, in this study, we compress an n-dimensional vector into m dimensions, as in $\boldsymbol{y} = \mathbf{A}\boldsymbol{x_0}$.

As summarized in Table I, 8192 elements were embedded into a single ciphertext. Therefore, the client packs the maximum number of plaintexts (vectors) into a single ciphertext and sends it to the server. Thus, the client packs slot_count$/m$ vectors into a single vector.

C. Reconstruction algorithm

HE is not suitable for comparison. Moreover, if the algorithm is complex, the number of homomorphic operations increases. Therefore, we reduce the reconstruction execution time by considering algorithms that do not require comparison operations and require fewer operations. This study estimates the data before compression by L_2 norm minimization using a generalized inverse based on these points. Because a dictionary is used, when the reconstruction algorithm is applied to (3), $encrypt(\boldsymbol{c}) = (\mathbf{A'}^\top\mathbf{A'})^{-1}\mathbf{A'}^\top \times encrypt(\boldsymbol{y})$. Thus, $encrypt(\hat{\boldsymbol{x}}) = \mathbf{D}((\mathbf{AD})^\top(\mathbf{AD}))^{-1}(\mathbf{AD})^\top \times encrypt(\boldsymbol{y})$ using (1). Therefore, we obtain the product of the product of the generalized inverse of dictionary $\mathbf{D}$ and $\mathbf{AD}$, and the encrypted low-dimensional matrix.

Implementation of matrix multiplication with HE. The method for reconstructing slot_count$/m$ $\hat{\boldsymbol{x}}$ is illustrated in

Figure 3. To reconstruct multiple $\hat{\boldsymbol{x}}$ simultaneously, we concatenate $\mathbf{D}((\mathbf{AD})^\top(\mathbf{AD}))^{-1}(\mathbf{AD})^\top$ with size n rows and m columns horizontally to create a matrix of size n rows and slot_count columns. In the CKKS scheme used in this study, performing a rotation operation that rotates the ciphertext slots by a specified number of steps to the left or right is possible. This is used in the implementation of the matrix product of the reconstruction algorithm.

Number of homomorphic operations. Here, we analyze the number of rotations and homomorphic operations in the reconstruction algorithm. We assume that the length of the vector before compression is $n = 2m$, that is, the compression ratio is 50%. If the number of rotation calls in Steps 2 and 4 in Figure 3 is denoted by num_rotation, it is proportional to m.

$$\begin{aligned}\text{num\_rotation} &= n(\log_2 m + \text{slot\_count}/m)\\ &= 2m(\log_2 m + \text{slot\_count}/m)\\ &= 2m\log_2 m + 2\text{slot\_count}\end{aligned}$$

We analyze the number of multiplications. The number of plaintext and ciphertext multiplications in Steps 1 and 3 in Figure 3 is denoted by num_mult, then num_mult $=$ $n(1 + \text{slot\_count}/m) = 2m + 2\text{slot\_count}$ times, and is proportional to m. Therefore, adopting a smaller m, that is, reconstructing multiple short $\hat{\boldsymbol{x}}$, reduces the number of homomorphic operations. For instance, rather than encrypting two vectors of length 4096, encrypting 512 vectors of length 16 to reduce the number of homomorphic operations in the reconstruction is better.

V. Evaluation

A. Experiment overview

We used the Microsoft SEAL library to implement the baseline and proposed method in C++. We used a dataset for both the baseline and proposed method. We also used K-SVD for dictionary learning. The following metrics were used:

- The size of the HE ciphertext sent from the client to the server.
- Accuracy in the logistic regression analysis.
- Execution time of the server- and client-side.

Here, the uncompressed case is described as the baseline; that is, baseline refers to a method that encrypts the multidimensional data on the client and then sends it to the database server.

B. Experiment setting

Datasets. The following three datasets were used in the experiment: The data were set such that the total number was 16 bits of n. The following conditions were used when performing the logistic regression.

- Wine quality dataset [17]: Predict whether the wine is red or white
  - Training data: 1292 cases
  - Dictionary learning: 323 cases
- NHANES III dataset [18]: Predict whether the person has high blood pressure
  - Training data: 3129 cases
  - Dictionary learning: 782 cases
- Adult dataset [19]: Predict whether income exceeds $50,000 per year
  - Training data: 6512 cases
  - Dictionary learning: 1628 cases

TABLE I: Parameters related to the CKKS scheme of SEAL

	`param1`	`param2`
Ciphertext length slot_count	4096	8192
Security	128 bit	
Level	2	4
Precision when decrypting	60 bit	
Precision of the real number value when decrypting	Integer part	20 bit
	Decimal part	20 bit

Experimental environment. We conducted our experiments using Raspberry Pi as the client. The client device has four cores, ARM Cortex-A72@1.8 GHz each, with 4 GB of RAM. The server machine has 10 cores, an Intel®Xeon®Gold 5115 CPU @2.40 GHz each, with 192 GB RAM.

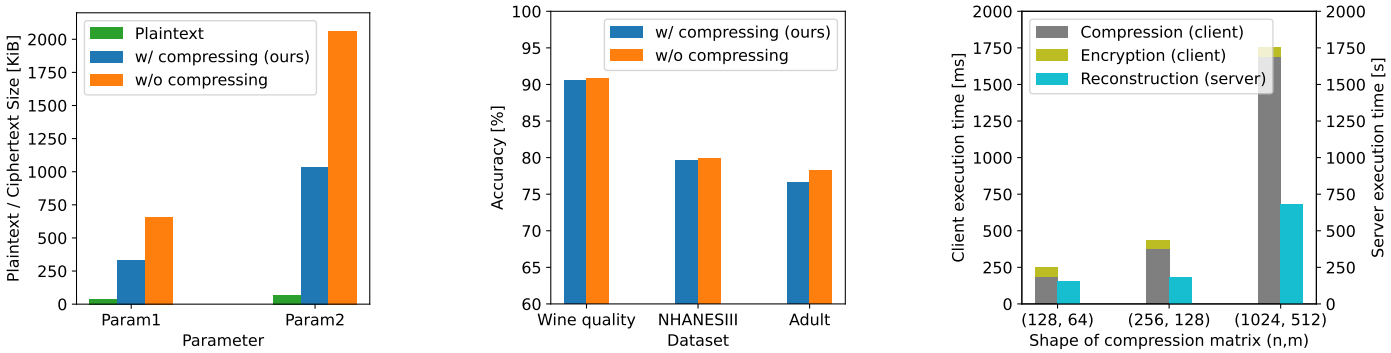
Parameters for HE. The parameters of the CKKS scheme from Microsoft SEAL [1] used in the experiment are listed in Table I. `param2` is set to run logistic regression. The modulus chain is set to $\{60, 40, 40, 60\}$ for `param1` and $\{60, 40, 40, 40, 40, 60\}$ for `param2`, allowing two and four homomorphic multiplications, respectively.

C. Communication volume

Compression before and after encryption. The simplest method involves compressing ciphertext files using the zip command. When encrypted $\boldsymbol{y}$ is compressed in the zip format, the size before compression is 1.1 MB, but after compression, it is 1.0 MB, exhibiting almost no size reduction. This is probably because compression using the zip format is more likely to compress data containing the same characters multiple times. HE ciphertexts do not have this characteristic; therefore, compression using the zip format is unsuitable.

Effect of compressed sensing. Figure 4(a) shows the communication volume between the client and server, that is, the size of the homomorphic ciphertext sent from the client to the server. `param1` encrypts 32 KiB of plaintext in a single encryption, and `param2` encrypts 64 KiB of plaintext in a single encryption. Compared to the baseline, the proposed method reduces the communication volume by approximately half. The ratio of the communication volume between the baseline and proposed method corresponds to the ratio of n to m; therefore, the values are irrelevant as long as the ratio of n to m is constant. For instance, the reduction in communication volume when $n = 1024, m = 512$ is the same as that when $n = 128, m = 64$. Additionally, increasing the value of n/m

[1] https://github.com/microsoft/SEAL



(a) Ciphertext size (b) Reconstruction accuracy (c) Execution time

Fig. 4: (a) In the proposed method, the communication volume of one ciphertext is reduced to approximately half compared to without compression in `param1` and `param2`. (b) A difference of approximately 0.3% – 1.6% between the baseline and proposed method in binary classification accuracy was observed. (c) Execution time at 50% compression and 32KiB plaintext with `param1`.

can reduce the communication volume, which also leads to a decrease in accuracy [20].

D. Reconstruction accuracy

We perform a logistic regression analysis on the baseline and reconstruction results of the proposed method using `param2`, and evaluate them by comparing the binary classification results. The shape of compression matrix $\boldsymbol{A}$ is $(16, 8)$. We first executed data normalization, and then binary classification was performed on 1024 cases of data, that is, slot_count$/m$.

Figure 4(b) shows the logistic regression analysis results for each column using the baseline and proposed method with a compression ratio of 50%. A difference of approximately 0.3% – 1.6% is observed between the baseline and proposed method. These results indicate that no significant error is present in the accuracy of the binary classification obtained by logistic regression analysis for the data with and without compression. The data reconstructed using the proposed method can be used for prediction.

E. Execution time of the server- and client-side

Shape of the matrix and the execution time. From Figure 4(c), the execution time decreases as the sizes of n and m decrease, even though the total amount of data handled remains the same. This is because the shape of compression matrix $\mathbf{A}$ is (n, m), and a positive relationship exists between the sizes of n, m and the time required for the matrix calculations. As explained in Section IV-C, the number of homomorphic operations in a reconstruction algorithm is proportional to m. Therefore, it can be assumed that it is faster and more suitable to reconstruct multiple short vectors instead of long vectors.

With or without compression. The execution times for the baseline (without compression) and proposed method with $n = 16, m = 8$ and `param2` are listed in Table II. We observe that even though the size of the ciphertext sent from the client to the server has decreased, the encryption time has increased compared with the baseline. In the proposed method, matrix $\boldsymbol{y}$ for each column is concatenated to form a single matrix $\boldsymbol{y}$ of length slot_count and then encrypted. Therefore, when using relatively small values, such as $n = 16, m = 8$, the concatenation process occurs many times, and we believe that the time is spent here. In fact, when we conducted an experiment using other data with $n = 256, m = 128$, we confirmed that the proposed method reduced the time taken for encryption as the size of ciphertext sent from the client to the server decreased. Moreover, during compression, the product of the $m \times n$ and $n \times 1$ matrices is calculated. From the results of the investigation of the execution time when varying the sizes of n and m, it can be inferred that the larger the sizes of n and m, the longer the time required for compression, even if the total length of the plaintext is the same.

TABLE II: Execution time per single vector $(n, m) = (16, 8)$

	Client		Server
	Compression	Encryption	Reconstruction
w/ compression (ours)	0.0195 [ms]	0.71 [ms]	1.128 [s]
w/o compressing	–	0.34 [ms]	–

VI. Limitations

In the proposed method, depending on the data distribution, dictionary learning may not be effective. For example, there may appear to be a great deal of variation in the data, with no similar patterns being seen. On the other hand, our method is probably effective for data such as electricity consumption, since the pattern of its appearance is fixed to some extent.

Additionally, as mentioned in Section V-E, the compression on the IoT and the reconstruction on the server takes time proportional to the shape of the compression matrix, and there is room for improvement in the computational overhead.

VII. Conclusion

In encrypted databases and encrypted regression analysis, which are examples of HE applications, the advantage is that only the aggregated/analyzed results can be made available to the data analyst; however, because the size of ciphertext is larger than that of plaintext, the communication volume between the client and server increases. Therefore, we proposed communication volume reduction through compressed sensing with dictionary learning. The experimental results demonstrated that when the compression ratio was set to 50%, the proposed method reduced the communication volume by 50% compared to when it was sent without compression. For accuracy, we performed a logistic regression analysis on the baseline and proposed method, and compared the binary classification results. This demonstrated that the data reconstructed using the proposed method can be used for predictions with negligible error.

Acknowledgment

This study was partly supported by JST CREST JPMJCR22M2.

References

[1] Nikola Samardzic, Axel Feldmann, Aleksandar Krastev, Srinivas Devadas, Ronald Dreslinski, Christopher Peikert, and Daniel Sanchez. F1: A fast and programmable accelerator for fully homomorphic encryption. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 238–252, 2021.

[2] Sangpyo Kim, Jongmin Kim, Michael Jaemin Kim, Wonkyung Jung, John Kim, Minsoo Rhu, and Jung Ho Ahn. Bts: An accelerator for bootstrappable fully homomorphic encryption. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 711–725, 2022.

[3] M Sadegh Riazi, Kim Laine, Blake Pelton, and Wei Dai. Heax: An architecture for computing on encrypted data. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 1295–1309, 2020.

[4] Marin Matsumoto and Masato Oguchi. IoT Device Friendly Leveled Homomorphic Encryption Protocols. In *2022 IEEE International Conferences on Smart Data (SmartData)*, pages 525–532. IEEE, 2022.

[5] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Annual Cryptology Conference*, pages 850–867. Springer, 2012.

[6] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, Xiaoqian Jiang, et al. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e8805, 2018.

[7] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*, pages 409–437. Springer, 2017.

[8] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *iEEE Access*, 10:30039–30054, 2022.

[9] Nazim Burak Karahanoglu and Hakan Erdogan. A* orthogonal matching pursuit: Best-first search for compressed sensing signal recovery. *Digital Signal Processing*, 22(4):555–568, 2012.

[10] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[11] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14:629–654, 2008.

[12] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993.

[13] Guangliang Chen and Deanna Needell. Compressed sensing and dictionary learning. *Finite Frame Theory: A Complete Introduction to Overcompleteness*, 73:201, 2016.

[14] Kobayashi Naoto and Okabe Takahiro. Efficient acquisition of images under multi-wavelength, multi-directional light sources by compressed sensing using dictionary learning, 2015.

[15] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[16] K-SVD implementation. https://github.com/takeshan/KSVD/tree/master.

[17] Wine quality. http://www3.dsi.uminho.pt/pcortez/wine/.

[18] LogisticDx: Diagnostic Tests for Models with a Binomial Response. nhanes3: NHANES III data. https://rdrr.io/rforge/LogisticDx/man/nhanes3.html.

[19] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[20] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.