

Retraction

Retracted: Cloud Storage Model Based on the BGV Fully Homomorphic Encryption in the Blockchain Environment

Security and Communication Networks

Received 5 December 2023; Accepted 5 December 2023; Published 6 December 2023

Copyright © 2023 Security and Communication Networks. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This article has been retracted by Hindawi, as publisher, following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of systematic manipulation of the publication and peer-review process. We cannot, therefore, vouch for the reliability or integrity of this article.

Please note that this notice is intended solely to alert readers that the peer-review process of this article has been compromised.

Wiley and Hindawi regret that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.


The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

- [1] J. Huang and D. Wu, "Cloud Storage Model Based on the BGV Fully Homomorphic Encryption in the Blockchain Environment," *Security and Communication Networks*, vol. 2022, Article ID 8541313, 9 pages, 2022.

Research Article

Cloud Storage Model Based on the BGV Fully Homomorphic Encryption in the Blockchain Environment

Jie Huang ^{1,2} and Dehua Wu^{1,2}

¹Hunan Provincial Engineering Research Center for Aircraft Maintenance, Changsha 410124, Hunan, China

²Changsha Aeronautical Vocational and Technical College, Changsha 410124, Hunan, China

Correspondence should be addressed to Jie Huang; huangjie918@163.com

Received 10 June 2022; Revised 29 June 2022; Accepted 2 July 2022; Published 20 July 2022

Academic Editor: Mohammad Ayoub Khan

Copyright © 2022 Jie Huang and Dehua Wu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Blockchain is a distributed time-series database. Based on the blockchain platform, this paper designs the framework model of cloud storage and designs the cloud storage model based on homomorphic encryption based on the operation process of the model. According to the applicability of underlying blockchain storage, the HElib library is established as the algorithm for data privacy protection. The BGV homomorphic encryption algorithm is used as the bottom layer of the algorithm, and the efficiency of the BGV homomorphic encryption algorithm is compared with that of the Gentry's bootstrap-based homomorphic encryption algorithm. It is proved that the BGV algorithm is more suitable for big data.

1. Introduction

At present, the blockchain cloud storage system is mainly used to store all the data on the chain, so each user on the chain should have the right to retrieve the data. The on-chain user, namely, the data owner, stores the data on the cloud server (CS). How can the nodes in blockchain quickly retrieve the corresponding data in the face of massive ciphertext is an urgent problem to be solved at present. Encrypting storage in cloud storage is a feasible solution. This paper designs a blockchain environment based on the BGV (2011 Brakerski, Gentry, Vaikuntanathan, a fully homomorphic encryption scheme, referred to as the BGV scheme [1]) fully homomorphic encryption cloud storage model, so that the cloud server can directly process the ciphertext. When users need corresponding data, they only send a request. The cloud server returns the processed data, and users get the processed plaintext data after decryption. Compared with other models, the cloud storage model proposed in this paper has high efficiency and simple operation and can ensure data privacy.

2. Methodology

2.1. Homomorphic Algorithm. Since this paper mainly uses blockchain as the basic platform, from the perspective of practicality, data retrieval, and checking whether data are linked, is the main demand in practical applications. Therefore, it is the main demand of the cloud storage system to select a homomorphic algorithm with high efficiency and suitable for retrieval. The following two important homomorphic algorithms will be analyzed [2].

From the perspective of applicability analysis of the basic status of homomorphism encryption, first of all, addition homomorphism and multiplication homomorphism are a part of the homomorphism encryption algorithm; this concept comes from recent algebra, Set $\langle G, * \rangle$ and $\langle H, * \rangle$ are two unrelated algebraic systems, $f: G \rightarrow H$ is a mapping, if $\forall a, b \in G$ will make $f(a * b) = f(a) * f(b)$, then F is called a homomorphic mapping from G to H [3]. The above formula is the multiplication homomorphic algorithm, and because the formula only meets the multiplication operation, not the addition operation, so it is a

partial homomorphic encryption operation. If both addition and multiplication can be satisfied, all the operations of addition and multiplication can be satisfied, which is called full homomorphic operation.

At present, there are many fully homomorphic encryption (FHE) algorithms. Gentry for the first time proposed a homomorphic encryption algorithm that can carry out both multiplication and addition operations, but its implementation in practical applications is low [4]. A BGV scheme designed a new homomorphic encryption construction technology, and the two schemes are described in detail as follows.

2.1.1. Gentry's FHE Scheme. Gentry's solutions are all "bootstrapping FHE solutions". Bootstrapping technology is the core of this technology, and the main idea is to reduce the noise in the ciphertext by processing its own decryption function, so that noise content is reduced to the range that can be correctly decrypted [5]. The specific implementation is the Recrypt_s operation in the Gentry's FHE scheme. The Recrypt_s operation performs noise reduction during the homomorphic decryption of the ciphertext. We set the message as m and the public key as pk_1 , then

$$c_1 = \text{Encrypt}_s(pk_1, m). \quad (1)$$

The encrypted ciphertext is c_1 . Assuming that the other public key is pk_2 and the decryption key after pk_2 is encrypted as sk_1 , set

$$\overline{sk_1} = \text{Encrypt}_s(pk_2, sk_1). \quad (2)$$

Recrypt_s operation:

$$\text{Recrypt}_s(pk_2, D_\epsilon, \overline{sk_1}, \overline{c_1}). \quad (3)$$

For each bit of c_1 , c_{1j} is

$$\overline{c_{1j}} = \text{Encrypt}_s(pk_2, c_{1j}). \quad (4)$$

It is written as

$$\overline{c_1} = \langle \overline{c_{11}}, \overline{c_{12}}, \dots, \overline{c_{1i}}, \dots \rangle. \quad (5)$$

Output:

$$c \leftarrow \text{Encrypt}_s(pk_2, D_\epsilon, \overline{sk_1}, \overline{c_1}). \quad (6)$$

In fact, the whole process can be regarded as the second encryption of message m . The public key pk_1 is used for the first encryption, and the public key pk_2 is used for the second encryption. The function Recrypt_s is used to decrypt the encrypted data, and the result of the second encryption is retained, that is, the Recrypt_s function can retain the results of the outer layer and decrypt the inner layer in the case of two layers of encryption.

2.1.2. BGV Scheme. BGV designs a new homomorphic encryption construction technology to ensure security and improve efficiency. The development process introduces dimension modulus specification, which enables ciphertext

refresh to effectively control ciphertext dimension and noise growth, and constructs a hierarchical homomorphic encryption algorithm. Different from the Gentry's homomorphic scheme, it does not require a bootstrap process [6].

Based on the encryption scheme based on GLWE (general learning with error), the scheme is extended to construct a fully homomorphic scheme with a hierarchical structure. The scheme is evaluated by using the parameter, namely, the depth L of the arithmetic circuit. The process of the BGV solution is as follows:

Step 1: input safety parameters λ , circuit depth L , and bit value b to represent the process as

$$\text{Setup}(l^\lambda, l^L, b), \quad (7)$$

where $b \in \{0, 1\}$ is mainly used to judge whether the encryption scheme is based on the LWE scheme or the R-LWE scheme.

Second, we calculate the parameters as

$$\text{params}_j \leftarrow \text{Setup}(l^\lambda, l^{(j+1)\mu}, b). \quad (8)$$

$j \in \{L, 0\}$ decreases step by step, and

$$\mu = \mu(\lambda, L, b) = \theta(\log \lambda + \log L). \quad (9)$$

Step 2: the key generation process is expressed as

$$\text{KeyGen}(\{\text{params}_j\}). \quad (10)$$

To loop j from L to 0 , we perform the following steps:

(1) We run the formula

$$\begin{aligned} s_j &\leftarrow \text{SecretKeyGen}(\text{params}_j), \\ A_j &\leftarrow \text{PublicKeyGen}(\text{params}_j, s_j). \end{aligned} \quad (11)$$

(2) We run the formula

$$s'_j \leftarrow s_j \otimes s_j \in R_{q_j} \left(\frac{n_j + 1}{2} \right). \quad (12)$$

s'_j is the tensor product of s_j with itself.

(3) set

$$s''_j \leftarrow \text{BitDecomp}(s'_j, q_j). \quad (13)$$

(4) We run the formula

When $j=L$, we do not perform this operation.

According to the calculation result in Step 2, the private key sk is s_j , and the public key pk is A_j and $\tau_{s_{j-1}}^n \rightarrow s_j$

Step 3: the encryption process is represented as

$$\text{Enc}(\text{params}, pk, n). \quad (15)$$

We make information m in R_2 , and we run $\text{Enc}(AL, m)$.

Step 4: the decryption process is

$$\text{Dec}(\text{params}, sk, c). \quad (16)$$

Assuming that the ciphertext c is encrypted under the public key generated by the private key s_j , we run $\text{Dec}(s_j, c)$.

Step 5: the process of performing homomorphic addition and multiplication is as follows:

- (1) The homomorphic addition process is as follows: add (pk, c_1, c_2) , so that the ciphertext c_1, c_2 are encrypted under the public key pk generated by the same private key s_j . Setting $c_3 \leftarrow (c_1 + c_2) \bmod q$, c_3 is the ciphertext under s_j ; the output ciphertext refresh result is $c_4 \leftarrow \text{Refresh}(c_3, \tau_{s_j}^n \rightarrow s_{j-1}, q_j, q_{j-1})$;
- (2) The homomorphic multiplication process is as follows: first, the ciphertext is multiplied by $\text{Mult}(pk, c_1, c_2)$ to obtain the new ciphertext, which is mainly generated by encrypting the public key pk generated by s_j under the private key. The new ciphertext obtained by multiplication is the coefficient vector c_3 of linear equation $L_{c_1, c_2}^{\text{long}}(x^2)$ under the private key $s_j' \leftarrow s_j \otimes s_j$, output: $c_4 \leftarrow \text{Refresh}(c_3, \tau_{s_j}^n \rightarrow s_{j-1}, q_j, q_{j-1})$.

Step 6: the ciphertext refresh process is represented as $\text{Refresh}(c_3, \tau_{s_j}^n \rightarrow s_{j-1}, q_j, q_{j-1})$. The input is the ciphertext c encrypted under the key s_j' , the auxiliary information $\tau_{s_j}^n \rightarrow s_{j-1}$ used for the key exchange, the current module q_j , and the next module q_{j-1} , which are divided into the following three steps.

- (1) Extension: set $c_1 \leftarrow \text{Powersof2}(c, q_j)$;
- (2) Analog to digital: $c_2 \leftarrow \text{Scale}(c_1, q_j, q_{j-1}, 2)$;
- (3) Key exchange: output ciphertext $c_3 \leftarrow \text{SwitchKey}(\tau_{s_j}^n \rightarrow s_{j-1}, c_2, q_{j-1})$, c_3 is the ciphertext encrypted by the private key s_{j-1} with the module q_{j-1} .

BGV is more efficient than homomorphic encryption, so it is more likely to be used in real scenarios.

The cloud storage system designed in this paper mainly determines the private key algorithm according to the application scenarios of the system. The current scheme uses the fully homomorphic encryption (FHE) algorithm library HElib to construct an FHE arithmetic model based on the BGV homomorphic algorithm. FHE arithmetic operations on arbitrary length integers and floating point numbers are implemented. In addition, some optimization methods, such as ciphertext packaging technology, are applied in the library.

2.1.3. Full Homomorphic Encryption Algorithm Library HElib. HElib is an effective homomorphic encryption software library that implements homomorphic algorithms based on BGV schemes. BGV is a completely homomorphic encryption scheme based on learning with errors (LWE) and independent of ideal lattices. The detailed process is described in Section 2.1.2. The main idea of the scheme is to shorten the ciphertext by using a mode dimension reduction technology, which greatly reduces the complexity of decryption and achieves the high efficiency of decryption when the ciphertext order increases [7]. In addition, directly applying the homomorphism algorithm to low-order operation is inefficient

and waste. Therefore, the efficiency of homomorphism is greatly improved by packaging multisegment ciphertext into a batch for packaging [8].

In addition to basic functions and algorithm optimization, the library also has some other useful functions, such as simple encryption functions [9]. The growth of noise in ciphertext is a big problem hindering its efficiency. Homomorphic addition causes the accumulation of noise. Homomorphic multiplication is the product of ciphertext noise on both sides, and its growth rate is much faster than addition. Therefore, in the HElib library, ciphertext refresh is mainly carried out after each homomorphic multiplication operation to achieve noise reduction. Theoretically, the noise growth rate of homomorphic multiplication is higher than that of homomorphic addition because the noise growth rate of homomorphic encryption addition is the sum of two ciphertext noises and the noise growth rate of homomorphic multiplication operation is the product of ciphertext noises. Therefore, after homomorphic multiplication, ciphertext refresh is required to reduce noise. Although the noise growth of homomorphic addition is relatively small, the increasing number of operations also requires ciphertext refresh to reduce noise. Since the bottom layer of the algorithm library is the operation in the logic circuit, and ciphertext refresh in the HElib library will consume a layer of modulus, let L_s be the total number of analog-to-digital conversion, the modulus in the modulus chain L_c , and the multiplication depth L . The main relationship of these three parameters is

$$L_c = L_s + 1, \quad (17)$$

$$L_s \approx 2 \left\lceil \frac{L}{2} \right\rceil. \quad (18)$$

From formulas (18) and (19), it can be concluded that

$$L_s \approx 2 \left\lceil \frac{L}{2} \right\rceil + 1. \quad (19)$$

In the HElib library, the larger the modulus value of the module chain is, the lower the efficiency of ciphertext homomorphism calculation. Therefore, the multiplication depth L should be reduced as much as possible in practical application. The cloud storage system described in this chapter is mainly applied to data retrieval. The complexity of the cloud storage system is equivalent to addition and meets system requirements.

2.2. Cloud Storage Model Based on the BGV Fully Homomorphic Encryption in the Blockchain Environment. Using the existing technology of cloud storage, combined with the actual characteristics of cloud on blockchain data, this chapter adopts the method of cloud (untrusted participants) storing the data uploaded by users. When uploading data, users perform homomorphic encryption to avoid risks during data transmission. The key is only owned by users, preventing data leakage during transmission. The interaction between users and the cloud storage systems consists of four processes: key distribution, data storage, data retrieval, and data retrieval.

2.2.1. Key Distribution. On the blockchain platform, users participate in the consensus process, and then the corresponding host node stores the transaction data on the chain. The transaction on each block will be sequentially stored in memory, and then it will be persistent, that is, these data will be written to the disk. Generally, level DB is used as a database with LSM tree as its storage structure. Since each user needs all the data on the storage chain, the pressure on a single node will gradually increase in practical applications, and the demand for the node host performance will also increase. Therefore, users will initiate the demand for data on the cloud. First, authentication is required between users and cloud servers [10]. Figure 1 shows the process of key distribution to facilitate subsequent data request by users. As can be seen from Figure 1, the cloud authentication center generates the public and private keys of the user and the server. The user has his own private key and the public key of the server, and the cloud server also has its own private key and the public key of the user.

The cloud authentication center generates the public and private keys of users and cloud servers. When users initiate cloud storage, they need to apply for the key from the cloud authentication center and store the public key of corresponding users on the cloud server for later ciphertext processing.

2.2.2. Data Storage. In this section, a relatively efficient homomorphic encryption algorithm is adopted to construct a cloud storage system. The system is mainly applied to blockchain, which encapsulates many encrypted transactions. Users mainly inquire whether the transaction exists, as shown in Figure 2. As shown in Figure 2, the main idea of uploading cloud data is to upload the block id, transaction hash, sender and receiver of the transaction, and transaction status on the cloud. Users can query the actual situation of the transaction and whether the transaction is on the chain.

Figure 3 shows the framework of CSS based on the storage units mentioned above. As can be seen from Figure 3, the blockchain platform is the actual use scenario of users. The client is the end of initiating requests for users, while the cloud service is the end of receiving requests from users. Other participants, such as blockchain managers, can also interact with the cloud server to initiate requests.

As shown in Figure 3, the steps of initiating storage on the cloud for user data are as follows:

Step 1: users initiate storage requests based on their own requirements.

Step 2: after receiving the request, the client uses the homomorphic encryption algorithm to encrypt data.

Step 3: ciphertext on the cloud. The data generated by the client based on the encryption algorithm is transmitted to the cloud server. Because the data is encrypted, no additional protection is required during data transmission.

Step 4: after the data is uploaded to the cloud data storage center, the data is stored in the rented storage space, and the public keys of the users are also stored in

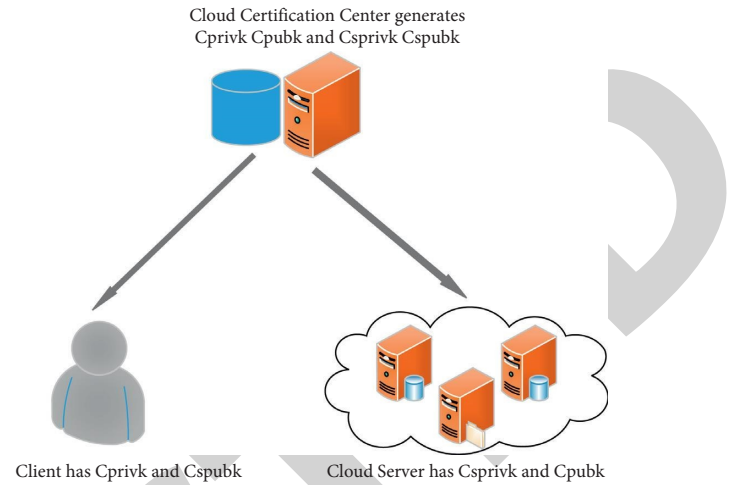


FIGURE 1: Key distribution process.

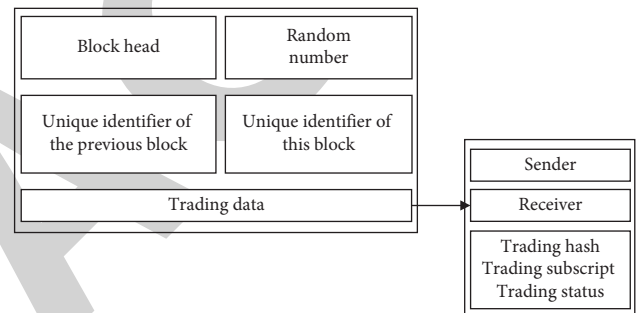


FIGURE 2: Blockchain uploads cloud data.

the public key libraries of the users. During data processing, users initiate data processing requests through server applications, such as compression and retrieval [11].

Step 5: the cloud storage server transmits the homomorphic encrypted ciphertext to the data processing module.

Step 6: the data processing module processes the data.

Step 7: the data processing module collects the processed data and returns the processed and sorted data to the server application.

Step 8: the user module loads the processed data.

Step 9: the homomorphic encryption module gets the processed data to decrypt the data to get plaintext.

Step 10: the decrypted plaintext to the user is returned.

The above steps are details of data storage on the cloud. The homomorphic encryption algorithm adopted in this chapter is the FHE algorithm library HELib, which is based on the calculation model of the BGV homomorphic algorithm.

2.2.3. Data Retrieval. Inverted index is a mapping structure often used in full-text retrieval [12]. In the cloud storage system designed in this chapter, users generally search for

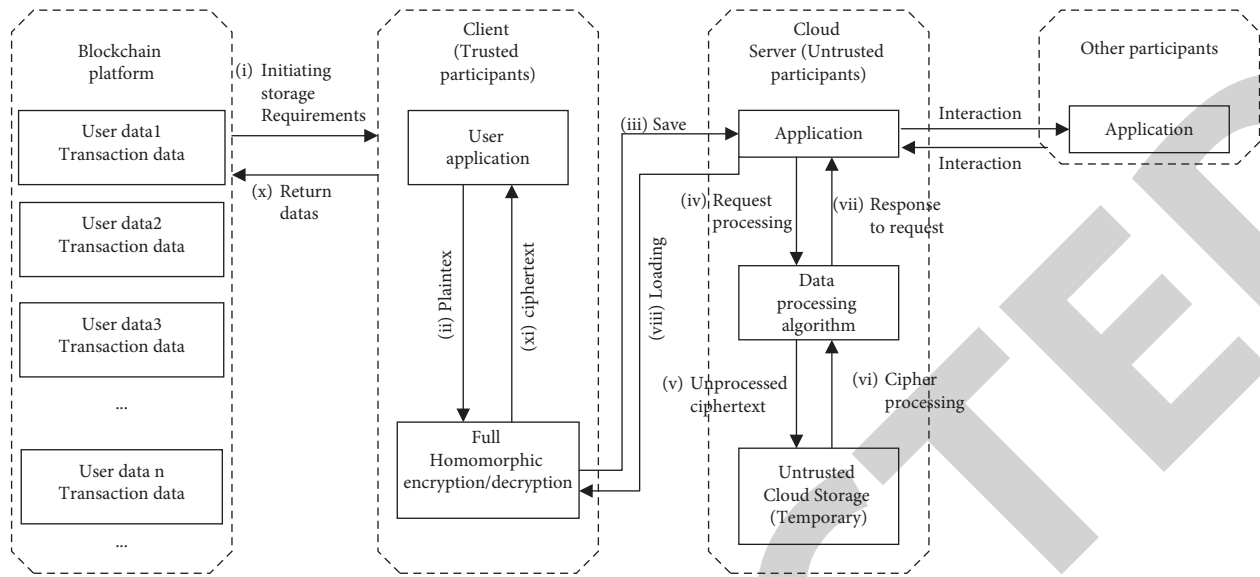


FIGURE 3: Cloud storage framework based on blockchain.

the existence of corresponding data and its corresponding attributes by entering keywords. However, as the amount of data increases, the complexity of data matching will increase. In addition, as the data of the designed system is uploaded through encryption, it is of great importance to improve the efficiency of retrieval and query, help users quickly locate target information, and reduce the difficulty of information acquisition.

Figure 4 is a schematic diagram of an inverted index query. As shown in Figure 4, the request module is used to receive the query's request and keywords and return the request results. In the inverted index structure, each word is followed by a corresponding linked list, in which the document number containing the word is stored. According to this structure, the document in which each word is located can be quickly queried. Applied to the blockchain platform, the data request is the hash of the transaction or the block, and the document in Figure 4 records the status of the transaction as well as sender and receiver information.

The inverted index model is combined with the cloud storage model designed in this chapter to obtain the data retrieval model. The detailed steps of the data retrieval process based on inverted indexes are shown in Figure 5. According to Figure 5, the detailed steps of the data retrieval scheme based on inverted indexes are as follows:

- Step 1: the user initiates a keyword retrieval request and uses the server public key to encrypt it.
- Step 2: the cloud server receives the encrypted keywords, decrypts them, and queries the index database for the existence of inverted indexes. If yes, go to Step 3. If no, go to Step 4.
- Step 3: ee return the results according to the retrieval process.
- Step 4: we establish the data inversion index according to keywords.

Step 5: the server verifies the validity of the retrieval results.

Step 6: we search keyword results that is returned.

Step 7: the retrieval results are encrypted with the user's public key and transmitted to the user.

Step 8: the user decrypts data.

The data retrieval process based on the inverted index can effectively improve the efficiency of data retrieval by establishing the index. Since the BGV homomorphism algorithm supports data retrieval, the data retrieval process described in this section is efficient and suitable for blockchain application scenarios.

2.2.4. Data Update. The cloud storage system designed in this paper is mainly based on the homomorphic encryption algorithm and stores the increasing data on the blockchain platform [13], so the cloud storage solution also needs to design appropriate data update strategy. The data update of the original symmetric and asymmetric encryption system always decrypts the original data, and then further updates the decrypted plaintext, which consumes a lot of resources and time in terms of implementation. In the cloud storage model based on homomorphic encryption, users generate data documents, encrypt them, upload them, and send an update operation request. After obtaining the request, the cloud server updates the ciphertext according to the operation request to obtain the updated ciphertext. The specific process is shown in Figure 6.

As can be seen from Figure 6, the specific steps of the data update process are as follows:

- Step 1: the user sends an update request to the server.
- Step 2: we encrypt the unupdated data to generate a data document.

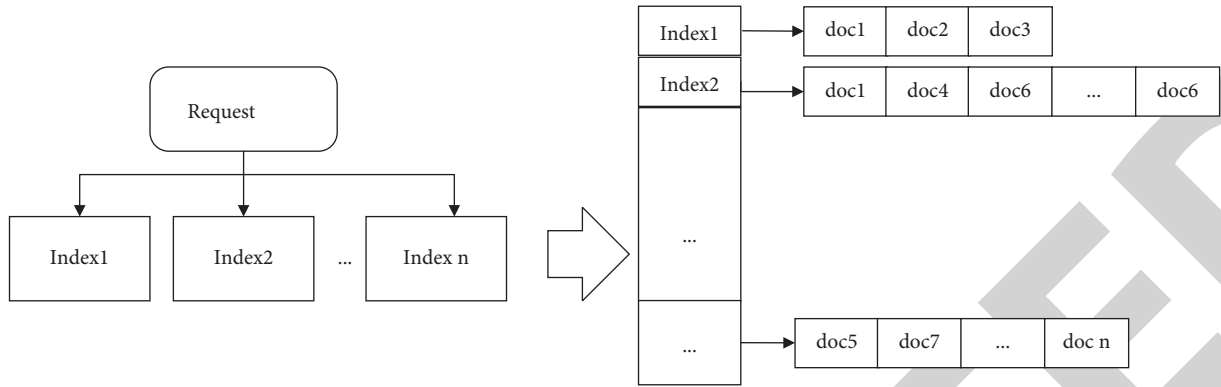


FIGURE 4: Data query flow of the inverted index.

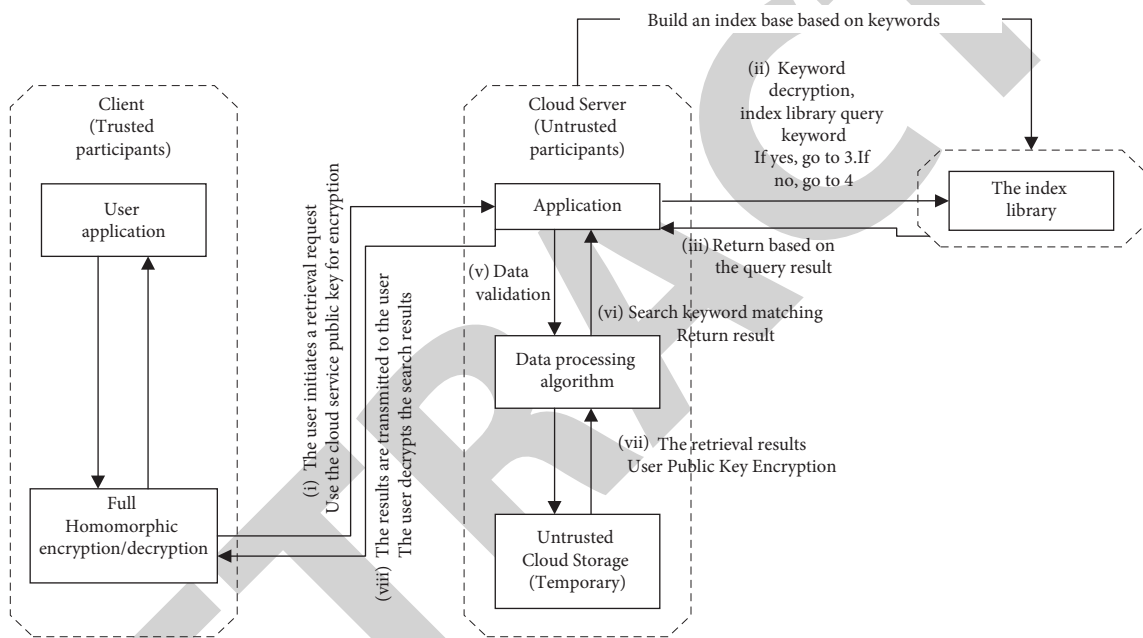


FIGURE 5: Data retrieval process based on the reverse sorting index.

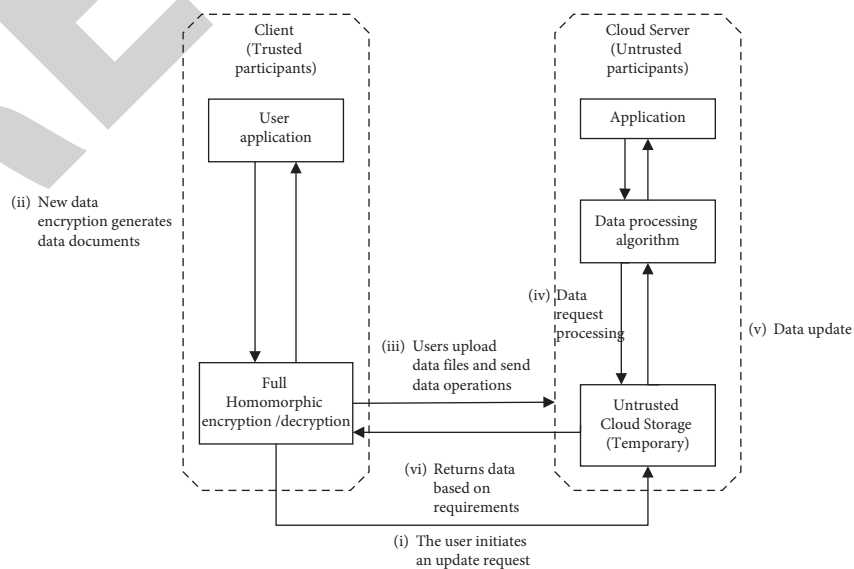


FIGURE 6: Data update process.

Step 3: after the encrypted document is generated, we upload the data to the cloud and send the data update request.

Step 4: the cloud sends an update processing request to the data processing module.

Step 5: the data processing module updates the data according to the operation and returns the updated data to the cloud server.

Step 6: the cloud server returns the updated data to the user as required.

Due to the time sequence and a large amount of blockchain data, continuous superimposed data will reduce efficiency in the actual user retrieval and the upload process. Therefore, the data update strategy effectively fits the user's usage scenarios and updates stored data according to user requirements, saving cloud server resources and avoiding waste.

3. Results and Discussion

3.1. Security Evaluation. As a distributed large-scale data storage system, the confidentiality of blockchain is the primary feature that needs to be guaranteed. Therefore, the most important feature of the user data stored on the chain in the cloud is data privacy. Different cloud storage systems have different security advantages and disadvantages. Generally speaking, the key points affecting data privacy mainly focus on the encryption algorithm, data transmission path, and key management.

Encryption algorithm is the basis of storage model privacy protection. The cloud storage system designed in this paper is a storage model based on homomorphic encryption. Only the sender has the encryption key of the data, and the data need to be encrypted before transmission. After being uploaded to the cloud, the data will be processed according to the demand, but the plaintext data will not be leaked all the time. The sender only needs to get the corresponding decryption of data which can be processed after the data, and data security is improved.

Data transmission path is also one of the important factors affecting data leakage. Generally, the data of cloud storage model is through the Internet communication network, which depends on the Internet environment, so the model may have the risk of information leakage. However, if the transmitted information is ciphertext, the risk of being stolen or monitored is the main risk. Therefore, it is necessary to adopt more secure network protocol in the transmission process.

Key management also affects data security in cloud computing environments. For the cloud storage framework with homomorphic encryption, the key management organization generates a key for each user. The key is encrypted only during data upload and is not leaked. Therefore, the complexity of the key management scheme is low. The security evaluation of the three cloud storage system models described above is summarized in Table 1.

3.2. Efficiency Assessment. Overall, the efficiency of the cloud storage system refers to the data encryption efficiency,

transmission efficiency, retrieval efficiency, and data update efficiency. In terms of design details, it includes the means to ensure data privacy, namely, the homomorphic encryption security algorithm.

3.2.1. Evaluation of Cloud Storage System Efficiency. Cloud storage systems, especially those on the blockchain platform, are most widely used for data retrieval. Different models have different retrieval schemes. For symmetric encrypted cloud storage components, such as Amazon S3, data retrieval requires decryption of ciphertext data before retrieval, which has low loss in terms of time and efficiency. The same is true for the public key encrypted cloud storage model. The idea of the core encryption algorithm of the homomorphic encrypted cloud storage model is that the processed plaintext data can still be obtained after the corresponding data is processed without data decryption. Therefore, this model supports the operation of ciphertext retrieval without decryption, and the operation of ciphertext directly is relatively efficient. However, in practical application, its accuracy is lower than that of plaintext retrieval because it needs to establish the data index to match ciphertext.

As data are constantly uploaded to the cloud, cloud storage allocation needs to be updated constantly. Therefore, data update is also an important part of the evaluation of cloud storage efficiency. In the cloud storage model with symmetric and asymmetric encryption systems, the main steps of data update are as follows: "data decryption, update, reencryption, and old data destruction". However, in the data update of homomorphic encryption, the client transfers the new data to the cloud, and the cloud directly updates the ciphertext data according to the data processing scheme, so the above steps are not required to operate it, just like the operation of plaintext. The update strategy is simple and efficient [14, 15].

3.2.2. Data Privacy Efficiency Assessment. The cloud storage system is built on the blockchain platform, which carries a large amount of data, and the search for and determination of the existence of data on blockchain is a big demand. Therefore, the complexity of data retrieval is an important feature of the homomorphic encryption algorithm in the system designed in this chapter. This section focuses on evaluating the encryption and decryption efficiency of the Gentry's FHE encryption algorithm and the BGV algorithm, as shown in Table 2.

It can be seen from Table 2 that the Gentry's scheme has low computing efficiency, and since data encryption and decryption will produce noise, data refreshing is required to reduce noise, and the time consumed by ciphertext refreshing is much larger than that of encryption and decryption. In addition, the ciphertext data after encryption expands greatly. As the amount of data increases, the time consuming becomes long and the efficiency decreases greatly. It is not practical to use this encryption method for massive data storage on the blockchain platform.

TABLE 1: Security comparison of cloud storage models.

	Encryption algorithm	Data transmission path	Key management
Symmetric encryption system Cloud storage model (Amazon S3)	Symmetric encryption easy to crack	Secure transmission channels are required	Key management is complex and easy to leak
Public key encryption system Cloud storage model	Asymmetric encryption difficult to crack	No secure transmission is required	Key distribution is complex and secure
Homomorphic encryption system Cloud storage model	Homomorphic encryption difficult to crack	Transmission in ciphertext does not require secure transmission	Simple and efficient key management, good security

TABLE 2: Allowable conditions of the Gentry's FHE algorithm.

Operation	Execution time (s)
Encryption	2.33
Decryption	0.12
Cipher refresh	27.28

BGV is a learning with error (LWE) scheme based on polynomial rings. Both plaintext and ciphertext are defined on rings, so homomorphic encryption and decryption on ciphertext is closely related to operations on plaintext rings. According to the operation efficiency and algorithm principle of BGV in Table 3, it can be seen that the BGV scheme has no bootstrap operation. Meanwhile, the scheme adopts the mode of single instruction and multiple data streams to package the plaintext of multiple channels and combine it into a ciphertext, so that the ciphertext data expansion growth rate meets the requirements of application.

3.2.3. Efficiency Evaluation of System Retrieval. Data retrieval on the cloud is based on the premise of storing a large amount of data. With increasing data, the running time of the blockchain-based cloud storage system is also changing, and its distribution is shown in Figure 7.

As shown in Figure 7, this paper designs a cloud storage system to store the data of blocks in blockchain. For a cloud storage system that does not use the inverted index scheme, the time spent in data retrieval gradually increases with the increasing amount of data, as shown in Figure 2. The relationship between the two is, approximately, proportional.

At the same time, Figure 7 also shows the data retrieval situation of the cloud storage system with the introduction of the inverted index. The curve shows a relatively smooth state in the follow-up, and the growth rate of time spent decreases with the increase of data volume. By comparing the two, it can be concluded that the retrieval scheme of the inverted index is introduced. In the process of increasing data, the time for users to retrieve data will be shortened correspondingly, which gradually improves the efficiency of retrieval.

3.3. Economic Performance Evaluation. Firstly, in terms of computing resources, different models have different resource consumption according to application scenarios.

TABLE 3: BGV algorithm operation.

Operation	Execution time (s)
Encryption	2.47
Decryption	0.57

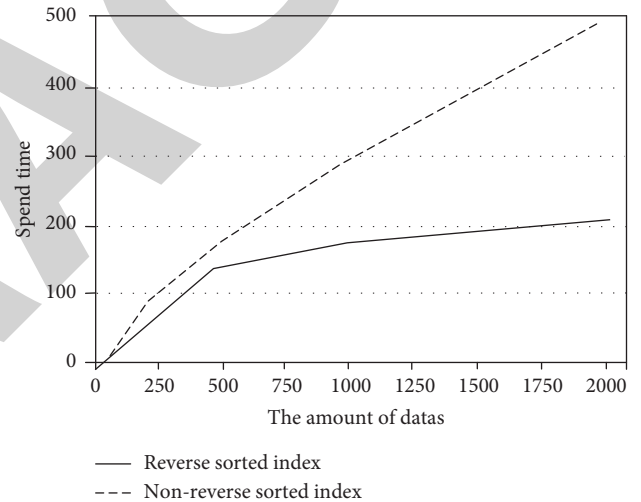


FIGURE 7: Retrieval performance with and without inverted indexes.

First of all, in terms of data retrieval, although the homomorphic encryption model is not as accurate as other models, its efficiency and consumption are lower. In addition, in terms of data update, the model abolishes the traditional basic process that can only be updated after decryption, greatly reducing the time and power of calculation. On the other hand, the amount of data server can carry is also limited, so it is necessary to set reasonable requirements and allocate computing resources. Secondly, in terms of technical input, the encryption algorithm and the security mechanism of homomorphic encryption are not fully mature in the field of ensuring cloud storage data security, and the input needs to be further increased.

4. Conclusion

This paper mainly designed the framework model of the cloud storage system, investigated the model-based

operation process on the market, designed the cloud storage model based on BGV fully homomorphic encryption. According to the applicability of the underlying blockchain system, after comparing the efficiency and practicality with the Gentry's homomorphic encryption algorithm, the BGV homomorphic encryption algorithm is used to protect data privacy in this study. In addition, since the application scenarios of the designed system mainly lie in the data state query and data retrieval, the inversion index algorithm is determined to design the retrieval scheme, and the key distribution, data storage, data retrieval, and data update processes are designed in detail. Comparing the efficiency of the privacy protection homomorphic algorithm BGV with the Gentry's FHE scheme in terms of operation time and ciphertext bloat, it is concluded that the BGV scheme is more suitable for big data.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Natural Science Foundation of Hunan Province in 2022: "Design and Application of Cloud Storage Security Architecture Based on Blockchain." (research funder: Huang Jie, Grant number: 2022JJ60091).

References

- [1] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from LWE," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 97–106, Palm Springs, CA, 2011.
- [2] C. Gentry, A. Sahai, and B. Waters, *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, attribute-based*, pp. 75–92, Springer, Berlin, 2013.
- [3] R.-L. Lagendijk, Z. Erkin, and M. Barni, *Encrypted Signal Processing for Privacy protection Conveying*, pp. 82–105, IEEE Signal Processing Magazine, 2013.
- [4] Z.-H. Mahmood and M.-K. Ibrahim, "New fully homomorphic encryption scheme based on multistage partial homomorphic encryption applied in cloud computing," in *Proceedings of the Annual International Conference on Information and Science*, pp. 182–186, Fallujah, Iraq, February 2018.
- [5] C. Gentry and S. Halevi, "Fully homomorphic encryption without squashing using depth-3 arithmetic circuits," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 107–109, Palm Springs, 2011.
- [6] K. Hariss, M. Chamoun, and A.-E. Samhat, "On DGHV and BGV Fully Homomorphic Encryption schemes," in *Proceedings of the 2017 1st Cyber Security in Networking Conference*, pp. 1–9, Rio de Janeiro, Brazil, January 2017.
- [7] C. Lupascu, M. Togan, and V. Patriciu, "Acceleration techniques for fully homomorphic encryption schemes," in *Proceedings of the 2019 22nd International Conference on Control Systems and Computer Science*, pp. 118–112, Bucharest, Romania, June 2019.
- [8] S.-M. Ghanem and I.-A. Moursy, "Secure multiparty computation via homomorphic encryption," in *Proceedings of the 2019 Ninth International Conference on Intelligent Computing and Information Systems*, pp. 227–232, Cairo, Egypt, March 2019.
- [9] R. Jain, S. Madan, and B. Garg, "Homomorphic Encryption over integers," in *Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development*, pp. 774–778, June 2016.
- [10] M. Ilic, P. Spalevic, and M. Veinovic, "Inverted Index Search in Data mining," in *Proceedings of the 2014 22nd Telecommunications Forum Telfor*, pp. 943–946, Belgrade, Serbia, February 2014.
- [11] X. Hongju, W. Fei, and W. FenMei, "Some Key Problems of Data Management in Army Data Engineering Based on Big Data," in *Proceedings of the 2017 IEEE 2nd International Conference on Big Data Analysis*, pp. 149–152, Beijing, China, October 2017.
- [12] L. Li, L. Yang, and Y. Di, "Distribution Characteristics of Transmission Line Galloping Events and the Analysis of Key Meteorological factors," in *Proceedings of the IEEE 3rd Conference on Energy Internet and Energy System Integration*, pp. 2360–2363, Changsha, China, April 2019.
- [13] M.-C. Nguyen and H.-S. Won, "Data Storage Adapter in Big Data Platform," in *Proceedings of the 2015 8th International Conference on Database Theory and Application*, pp. 6–9, Jeju, Korea (South), March 2015.
- [14] S. Verma and R.-A. Satao, "A Survey on the Impact of Economies of Scale on Scientific communities," in *Proceedings of the 2015 International Conference on Advances in Computer Engineering and Applications*, pp. 722–726, Ghaziabad, India, July 2015.
- [15] R. Liu, W. Sun, and W. Hu, "Planning of geo-distributed cloud data centers in fast developing economies," in *Proceedings of the 2018 20th International Conference on Transparent Optical Networks*, pp. 1–4, Bucharest, Romania, September 2018.