IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 31, 2019, accepted May 4, 2019, date of publication May 8, 2019, date of current version May 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2915576

Secure Cloud Storage Service Using Bloom Filters for the Internet of Things

JUNHO JEONG[1], JONG WHA J. JOO[1], YANGSUN LEE[2], AND YUNSIK SON[1]
[1]Department of Computer Science and Engineering, Dongguk University, Seoul 04620, South Korea
[2]Department of Computer Engineering, Seokyeong University, Seoul 02713, South Korea

Corresponding author: Yunsik Son (sonbug@dongguk.edu)

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2017R1D1A3B03029906), and this research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (No. 2016R1A2B4008392), and this work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2018R1A5A7023490).

ABSTRACT Today, the Internet of Things (IoT) is used for convenience in everyday life in many areas. Owing to the fact that the data collected from the IoT are generated in large quantities, cloud computing is inevitably used to store and analyze the data. However, cloud storage is not owned by the user, so it is unreliable. Verifying the integrity of data collected in an IoT environment and stored in a cloud has two problems: a large amount of data needs to be verified, and the data verification should be done directly on the IoT device. Many methods for data integrity verification use trusted third parties and devices that provide sufficient resources. However, it is difficult to directly apply existing research to the IoT devices that have limited resources. This paper proposes a secure cloud storage service for an IoT environment that is based on a provable data possession model and uses Bloom filters. The experimental results showed that the proposed method saves time and has no significant differences in the verification rate with existing methods, even though the Bloom filter causes false positives. Therefore, the proposed service can effectively process a large amount of data generated in an IoT environment.

INDEX TERMS Access control, computer security, cryptography, data security, data storage systems, distributed computing, Internet of Things.

I. INTRODUCTION

The continuous development of information communications technology (ICT) has led to an Internet of Things (IoT) environment that targets Internet connections for all devices. IoT technology enables diverse devices, including small sensors in a network, to have access to the Internet. Most IoT services collect data through sensor technology and transmit data to a cloud server over the Internet. The server analyzes the data by using artificial intelligence technology, generates information, and provides various services based on this information. Because IoT collects large amounts of data, cloud computing is inevitably used to store and analyze the data. As shown in Fig. 1, various smart services based on IoT services have emerged in diverse industries, such as healthcare, transportation, and home appliances [1], [2].

The basic operations of IoT-based services are as follows: Sensors are installed in objects to detect environmental information, and the detected data are sent through the Internet to a server for analysis. The server receives and analyzes the data to extract meaningful information, which is defined as data necessary to provide a specific service. Information can be acquired according to the data analysis method to provide users with useful services. Fig. 2 illustrates the IoT process, which consists of data collection and transmission, communication, data analysis, and services. In other words, IoT services are not only for detecting information but also a communication service for data transmission, a cloud service for storing and managing information generated from massive data, an artificial intelligence technology for data analysis, and control technology to control object devices. In recent years, various studies have been conducted on collecting data via sensor technology for analysis and processing [3]–[5].

For a system to be user-friendly, correct data analysis is needed. This requires the data collected from sensors to be well-managed. Based on the collected data and information generated through analysis, appropriate services can be provided to users. However, if this information disappears or

The associate editor coordinating the review of this manuscript and approving it for publication was Shuiguang Deng.

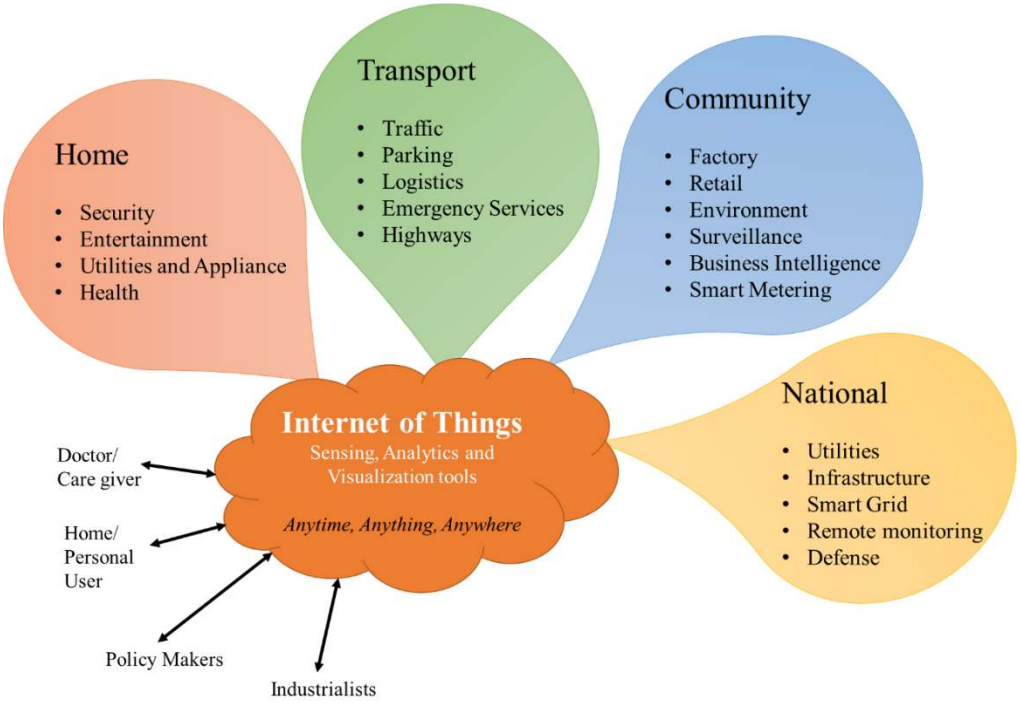

FIGURE 1. Various sample domains of internet of things service.

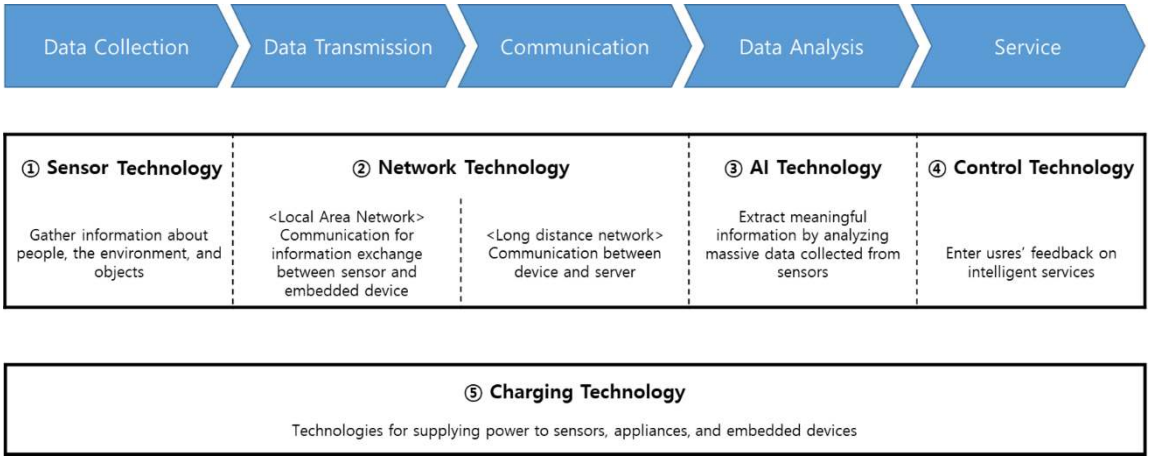

FIGURE 2. IoT services processes and key technologies.

is corrupted, providing users with the right services will be difficult.

For example, the smart home system is a representative IoT service. Home appliances connected to the Internet produce information and transmit it to other objects and people in a residential environment with an established wired/wireless communication network. The smart home system aims to improve the quality of life of residents by determining and predicting their demand and 2making decisions with a certain level of automation.

Smart homes involve various fields and have expanded in application in recent years. Smart home solutions and services (including cloud and big data on the backend) connect and control multiple smart devices, including home appliances, lighting, energy management, network, security, HVAC (heating, ventilating, and air conditioning) and home entertainment. In the future, smart homes will also include home-use robots and virtual reality.

However, if the control of various devices in a smart home refers to the wrong data, this may not only be inconvenient to users but also life-threatening if the wrong service is provided. For example, if incorrect data are recorded by an HVAC system when the temperature is normal, the smart system will attempt to reduce the temperature of the living space, which can lead to false control.

Therefore, data collected from IoT services and the extracted information should be managed securely. IoT services generate very large amounts of data that must be stored

TABLE 1. Statistics of data corruption incidents.

System name	System file corruption	Metadata corruption	Block corruption	Misreported corruption
Hadoop 1.x	15	11	46	4
Hadoop 2.x(YARN)	1	0	7	0
HDFS 1.x	17	7	23	7
HDFS 2.x	8	0	22	10

in a cloud. However, cloud storage services are basically untrusted storage. There is a risk that data stored in a cloud may disappear or be tampered with. The service is not directly managed by the user; the user only stores and retrieves data from the cloud service provider. Therefore, the data in the cloud storage may be corrupted or lost, or the server may go down and the user will be unable to access the necessary data. Wang *et al.* analyzed data corruption incidents in a typical cloud storage system [6]. As indicated in Table 1, the data from cloud storage were not reliable. In the table, YARN stands for "yet another resource negotiator," and HDFS stands for "Hadoop distributed file system."

This paper proposes a secure cloud storage service for an IoT environment that is based on an effective provable data possession scheme using a Bloom filter, which is a time- and space-efficient data structure that allows for some errors [7].

The Bloom filter is represented by an array of m bits for set $S = s_1, \ldots, s_n$ of n elements. The bits of all arrays are initialized to zero. The filter uses r independent hash functions $h_1, \ldots, h_r$. Each element s of set S is set to 1 in the array where $h_1(s), \ldots, h_r(s)$ are located. The same position may be repeatedly set to 1, but a position set to 1 cannot be reset to 0. If element a belongs to set S, this can be determined by checking whether or not every position bit of $h_1(a), \ldots, h_r(a)$ is 1. If all of the position bits identified in the array are 1, a belongs to set S. However, a Bloom filter operates according to a hash function, so false positives may occur due to a collision. This is an error saying that an element belongs to a group even though it is not actually a member. On the other hand, the Bloom filter has a feature that no false negatives occur, where an element included in the group is not a member. For this reason, the Bloom filter is a very effective data structure for confirming that an element is a member of a group if it treats only false positives well.

This paper is organized as follows. Section II provides an overview of related provable data possession schemes considered in this study. Section III introduces the proposed scheme. Section IV presents an analysis of the experimental results. Finally, Section V presents the conclusion.

II. RELATED WORKS

Cloud storage services are not only available to end users in a similar format to online shared folders but also provide storage in a variety of forms as needed. As presented in Table 2, storage services are shared for reduced data traffic and storage space efficiency, and data are stored unencrypted [8].

TABLE 2. Online storage service provider.

Name	Protocol	Encrypted transmission	Encrypted storage	Shared storage
Dropbox	proprietary	yes	no	yes
Box.net	proprietary	yes	yes (enterprise only)	yes
Wuala	Cryptree	yes	yes	yes
TeamDrive	many	yes	yes	yes
SpiderOak	proprietary	yes	yes	yes
Windows Live Skydrive	WebDAV	yes	no	yes
Apple iDisk	WebDAV	no	no	no
Ubuntu One	u1storage	yes	no	yes

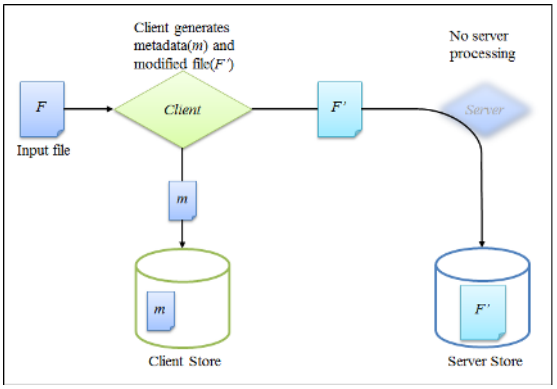

FIGURE 3. Preprocessing of PDP technique and data store method.

Services that provide storage other than online storage encrypt and store data but do not provide information about how data are stored internally. The problem is that it cannot be known whether the data are shared internally or tampered. For this reason, various studies have been conducted to verify the possession of data in unreliable storage, such as in cloud storage services [9], [10].

Li *et al.* proposed Secure Untrusted Data Repository (SUNDR), which is a network file system that securely stores data on untrusted servers [11]. SUNDR uses hash trees and chains to prevent fork attacks and ensure that data are not corrupted. It then verifies data stored in an untrusted third server through indicators such as provable data possession (PDP) and proof of retrievability (PoR) at the same time without retransmission of the actual data.

Ateniese *et al.* proposed a PDP scheme [12] that consists of two processes. The first is a preprocessing procedure, as shown in Fig. 3. Metadata m are generated with RSA-based homomorphic verifiable tags (HVTs) before the data are stored, and the metadata F are added to the file F to be stored in the server F' without any other operations being performed. The metadata are kept by the user and used for comparison in the next verification stage.

In the second process, the client verifies the actual ownership of the data stored in the server, as shown in Fig. 4.

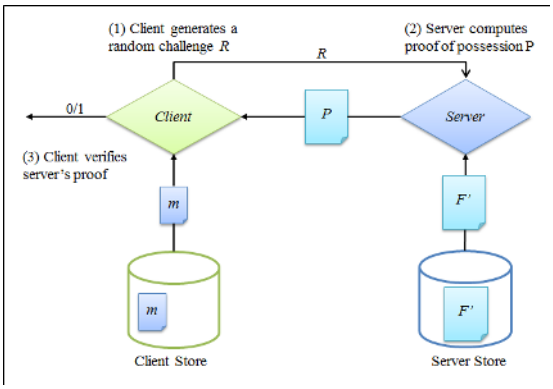

FIGURE 4. Data possession verification of PDP.

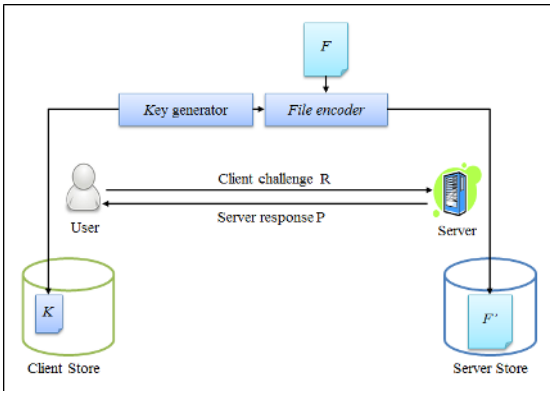

FIGURE 5. POR system model.

If the ownership is verified, the client generates a random number R and sends it to the server; the server sends the random number P to the client with the attached metadata for the file that received the verification request. The client receiving the result from the server checks the final result against the proof of ownership of the data by comparing it with the previously stored metadata.

PDP can be used to prove probabilistic integrity rather than a survey of all the data, as evidenced by the data ownership verification test method. The client can prove ownership of the data without storing the actual data. Ateniese *et al.* proposed a PDP scheme that considers active storage [13]. However, it only supports basic block operations with limited functionality and does not support block addition operations.

Erway *et al.* proposed dynamic provable data possession (DPDP), which is an expansion of PDP [14]. The proposed technique provides an authenticated dictionary with an RSA tree to help efficiently delete, modify, and add stored data.

Fig. 5 shows the POR scheme proposed by Juels *et al.*, which works similar to the PDP scheme [15]. However, it differs from PDP in that it does not store the metadata generated from the file but encrypts it through data encoding, stores it in the server, stores the key used for encoding, and uses the key when the metadata are verified in the future.

The user can verify the integrity of data stored in the server by using some requested information without downloading the entire data from the file. This is a very important factor in evaluating the integrity of large amounts of data or files. Bowers *et al.* used POR for deriving theoretical implementation results [16]. However, POR does not support the verification of highly variable data, and decoding operations performed during verification require a relatively large amount of resources.

Shacham and Water presented an improved POR scheme that can ensure complete security against attacks by malicious users [17]. Two types of techniques have been proposed. First, the Boneh–Lynn–Shacham (BLS) signature uses a bilinear map for verification and a signature as a group member of an elliptic curve. It is a public verification method with a short query and response [18]. Second is a private verification technique that uses relatively long queries and short responses. Both schemes aggregate queries and responses based on homomorphic attributes but require complicated computation.

Wang *et al.* proposed a security model that guarantees the integrity of stored data through public verification in a cloud environment. However, the external verifier does not guarantee the privacy of the user's data. For example, there is a potential threat that the user data reveal important information to the verifier [19], [20].

HAIL is a widely available scheme that applies an integrity layer to the cloud environment to extend the basic advantages of the redundant array of inexpensive disks (RAID) [21]. This scheme improves the message authentication code (MAC) to an integrity-protected error correcting code (IP-ECC) for increased security and efficiency and uses a verification approach similar to the POR technique. This technique can be applied to multiple server environments. However, the problem with RAID-based schemes is that one file must be converted to one distinct segment and stored in one server.

Zhu *et al.* proposed a cooperative provable data possession (CPDP) model in which data are verified in a multi-cloud storage environment, and a hierarchical hash index is applied to the PDP [22]. This scheme is cost-effective regarding computation and communication and has the advantage of being able to respond to various security attacks through continuous auditing. Yang and Jia proposed a cryptographic algorithm-based audit protocol to solve the data privacy problem where the auditor cannot recover the data block through an audit and neither the information used in the verification nor the contents of the original data are exposed [23]. Recently, various studies have been conducted to verify the data integrity of a cloud environment with regard to IoT. Liu *et al.* proposed a secure IoT data storage audit protocol based on Yang and Jia's technique and analyzed its performance [24]. They also conducted experiments to find the optimal values and outperformed Yang and Jia's technique. Sai *et al.* used a third-party proxy computing platform with fully homomorphic encryption to verify data integrity [25]. However, most of the existing proposed schemes generate metadata by using a

public key scheme such as RSA, pairing-based cryptography, and homomorphic encryption. Thus, a relatively large amount of resources is used for generating metadata and verifying data possession.

Aditya *et al.* proposed using a Bloom filter for each block of data stored in a server to verify data possession [26]. The advantage of this method is that using a Bloom filter allows more space for computation and metadata compared to existing techniques. However, if a single file has 1000 blocks and the positive error rate of the Bloom filter is 0.0001, the reliability of the verification probability for one file is positive for all blocks; if it is not found, the reliability is only about 90%. In addition, during the data ownership verification, the entire file is downloaded from the server, and the encrypted Bloom filter has to be decrypted.

Thus, a secure cloud storage service requires a technique that can effectively verify a large amount of data generated in an IoT environment and that does not cause data privacy problems. Many existing studies have used a public key based on a cryptographic algorithm to encrypt plaintext for checking by a third party, which has relatively high computation costs. To solve this problem, we propose a provable data passion scheme that uses a Bloom filter instead of encrypting plaintext as the public key for data verification.

III. CLOUD STORAGE SERVICE BASED ON PROVABLE DATA POSSESSION USING BLOOM FILTERS

For proprietary verification techniques of data in unreliable storage, the metadata are generated and recorded before the data are stored. The user then performs a query to verify the data stored in the server, which returns the corresponding result. The user can compare the data returned from the server and the metadata recorded when the data were stored to determine whether the stored data were altered or lost.

The essential features of this system are the size of the query used for verification, the time to generate the query, the time to evaluate the response received from the server, the time for the server to generate the query response, and the size of the query response. The time required to generate the metadata is relatively insignificant because it only occurs when the initial data are stored.

Data in a cloud storage environment are divided into blocks and stored in the server. These blocks may be physically stored in the same server or in physically separate servers. Therefore, some blocks constituting the data can run into problems when the data are shifted, modulated, or lost. Table 3 presents the notation used in the proposed data possession scheme.

In the proposed method, to prove ownership of the server, the data in cloud storage are divided logically into n blocks, and each block is set as a group member of the Bloom filter. Because the proof of data ownership based on PDP is probabilistic, if the positive error rate of the Bloom filter is set to the same level as 0.0001, it will not have a significant influence on the verification of the data ownership.

TABLE 3. Notation for data possession scheme.

Variable	Definition
K_{priv}	Master key of data owner
K_{pub}	Public key of data owner
m	One block of data
M	Data to store on the server
h	Pseudo-Random Function
n	The number of blocks that constituting the data
t	Number of sampling blocks to be verified
ρ	Block tempering probability
s	Number of sectors in each block
Bn	List of blocks to be verified
BF	Bloom filter
P_f	False positive rate
IntegrityVerifyIndex	Data possession verification index
Verifier	Data possession verification auditor
Prover	Data possession verification a subject to audit

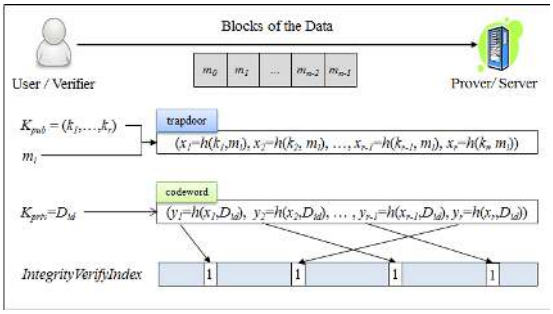

FIGURE 6. The pre-process for provable data possession using Bloom filter.

The proposed method can be divided into preprocessing and integrity verification. During preprocessing, the data owner creates a trapdoor with the block and its public key instead of keywords, such as a searchable encryption method from the data, before the data are stored on the server. Then, a code word is generated with a private key (e.g., a unique identification of data only known to the owner or a trusted third party), and a Bloom filter is set to generate a data ownership verification index. This index is stored by the owner or a trusted third party. Also, if the index is constructed using multiple Bloom filters with high false positive, it will be more effective in terms of security [27].

Fig. 6 illustrates the preprocessing of the proposed scheme. The user stores $M = \{m_0, m_1, \ldots, m_{n-1}\}$ consisting of n data blocks in the server for data ownership verification. When the data are stored, the user calculates the number of blocks of data M to be stored and generates a Bloom filter with the number of blocks and positive error rate.

For each block of data, the trapdoor $= \{x_1, x_2, \ldots, x_r\}$ is generated with the public key K_{pub}. This is used to generate the Bloom filter for generating the data ownership

verification index (*IntegrityVerifyIndex)*. The codeword $= \{y_1, y_2, \ldots, y_r\}$ is calculated from the trapdoor and its secret key K_{priv}, and the position corresponding to the codeword result in the Bloom filter is set to 1. This operation is repeated from block m_0 to m_{n-1} to determine the data ownership verification index from the Bloom filter.

The preprocessing is similar to creating an index of a document in searchable encryption, where each block of data acts as a keyword contained in the document. However, in a searchable encryption scheme, the key used to generate the trapdoor cannot be exposed because it is a secret. In the data ownership authentication scheme, the key for generating the trapdoor may be exposed to the outside as public. When a user verifies future data, even if a malicious user creates a trapdoor with a public key, it is impossible to create a trapdoor corresponding to the blocks being verified if the original data are not possessed.

The data possession verification process is as follows. First, an auditor such as a data owner or trusted third party randomly generates numbers of blocks of data for integrity verification and sends them to the server. Second, the server creates a trapdoor for each block requested by the auditor by using the public key of the data owner, sets it in one Bloom filter, and sends the Bloom filter to the auditor as a trapdoor that responds to the query. Third, the auditor can calculate the codeword by using the received trapdoor and obtains the verification result by referring to the data possession verification index.

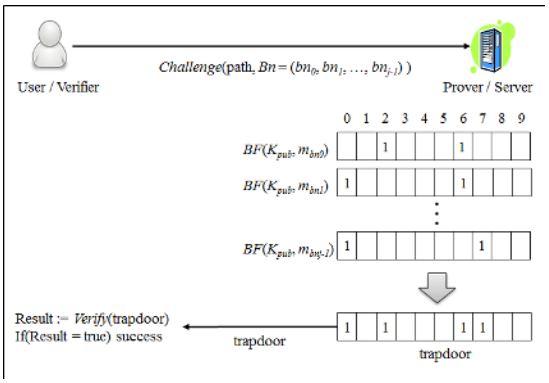

FIGURE 7. The verification process for provable data possession using Bloom filter.

Fig. 7 represents the verification process. The auditor sends *Challenge*, which includes a proven file path and selected random list of blocks $Bn = \{bn_0, bn_1, \ldots, bn_{j-1}\}$, to the server. The server that receives *Challenge* from the auditor generates the Bloom filter *BF* by using an already known false positive rate and the public key. When all trapdoors corresponding to *Bn* overlap and the trapdoor is generated, it is transmitted to the auditor to verify possession of the data through the *Verify()* operation. If the result is true, there is no abnormality. If it is false, it indicates that the data have been modified.

The *Verify()* operation computes the codeword y by using a pseudo-random function made up of the private key D_{id}, which is an integrated trapdoor $\{x_0, x_1, \ldots, x_{rt}\}$ matched to each block received from the server. The y-th position of the calculated index is confirmed to be set to 1; if any is set to 0, the evaluation result is returned as false. On the other hand, when the corresponding position of the index is confirmed to be set to 1 for all codewords calculated from the trapdoor, there is determined to be no forgery or tempering, and a positive value is returned.

TABLE 4. The influence of false positive rate on probability of data corruption detection.

ρ	P_f	t	P_d	P'_d	P_d-P'_d
0.1	0.0001	10	65.13%	65.09%	0.04%
0.1	0.001	10	65.13%	64.74%	0.39%
0.1	0.01	10	65.13%	61.06%	4.07%
0.1	0.1	10	65.13%	0.00%	65.13%
0.1	0.0001	20	87.84%	87.82%	0.03%
0.1	0.001	20	87.84%	87.57%	0.27%
0.1	0.01	20	87.84%	84.84%	3.01%
0.1	0.1	20	87.84%	0.00%	87.84%
0.1	0.0001	30	95.76%	95.75%	0.01%
0.1	0.001	30	95.76%	95.62%	0.14%
0.1	0.01	30	95.76%	94.09%	1.67%
0.1	0.1	30	95.76%	0.00%	95.76%
0.1	0.0001	40	98.52%	98.52%	0.00%
0.1	0.001	40	98.52%	98.45%	0.07%
0.1	0.01	40	98.52%	97.70%	0.82%
0.1	0.1	40	98.52%	0.00%	98.52%
0.1	0.0001	50	99.48%	99.48%	0.00%
0.1	0.001	50	99.48%	99.46%	0.03%
0.1	0.01	50	99.48%	99.10%	0.38%
0.1	0.1	50	99.48%	0.00%	99.48%

And the influence of false positive rate on probability of data corruption detection is shown in Table 4. In generally, the probability of data corruption detection is expressed as $1-(1-\rho)^t$ in the provable data possession schemes. P_d is the probability of detection according to this formula, and P'_d is the probability with the false positive rate in Table 4. It shows that the closer the false positive rate to the corruption rate, the lower the detection rate. However, as the false positive rate is lower and the number of sampling is higher than the data corruption rate, the false positive does not significantly affect the data corruption detection.

TABLE 5. Comparison of POR/PDP schemes for one file consisting of N blocks.

Scheme	Type	CSP Comp.	Client Comp.	Comm.	Prob. of Detection
PDP	*HomT*	$O(t)$	$O(t)$	$O(1)$	$1-(1-\rho)^t$
SPDP	*MHT*	$O(t)$	$O(t)$	$O(t)$	$1-(1-\rho)^{ts}$
DPDP-1	*MHT*	$O(t \log n)$	$O(t \log n)$	$O(t \log n)$	$1-(1-\rho)^t$
DPDP-2	*MHT*	$O(t \log n)$	$O(t \log n)$	$O(t \log n)$	$1-(1-\rho)^{\Omega(n)}$
CPOR-1	*MomT*	$O(t)$	$O(t)$	$O(1)$	$1-(1-\rho)^t$
CPOR-2	*MomT*	$O(t+s)$	$O(t+s)$	$O(s)$	$1-(1-\rho)^{ts}$
IPDP	*MHT*	$O(t)$	$O(t)$	$O(t)$	$1-(1-\rho)^t$
T.Aditya's	*BF*	$O(t)$	$O(t)$	$O(t)$	$1-(1-(\rho-(1-(1-P_f)^t))^t$
Proposed	*BF*	$O(t)$	$O(t)$	$O(t)$	$1-(1-(\rho-P_f))^t$

The proposed scheme is relatively efficient compared to the existing RSA-based data possession verification scheme. This is because, the server does not perform any operations when storing data but merely generates a Bloom filter for blocks being queried during the data possession verification process. In addition, even when the value of the trapdoor transmitted from the server to the auditor is exposed, no information can be inferred about the verification index or the data stored by the data owner.

In other words, there is no way to deduce the value or key of the data from this information, so it is safe from attack by a malicious user. With regard to the communication time between the server and auditor, it takes a fixed amount of time to return the trapdoor made of the same size as the Bloom filter, regardless of the number of data blocks being sampled for the stored data. Finally, the auditor can use simple hashing operations for verification, so the computation time is relatively efficient.

Table 5 compares the performance of the proposed scheme with that of the existing data ownership verification schemes for a single file consisting of n blocks. t is the number of blocks used for sampling during verification, and s is the number of sectors in each block. ρ denotes the probability of block corruption, and P_f denotes the false positive rate of the Bloom filter when used.

The proposed method did not show much difference with the existing techniques in terms of the operations of the cloud service provider, client, and between the server and client. The detection probability seemed to be lower because of false positives. However, if the Bloom filter is used for verification, it has an advantage compared to existing methods in terms of the time for the $O(t)$ operation. As mentioned previously, if the false positive rate is set to a very small value, it will not have a large effect on the probabilistic model for possession verification.

Finally, the additional consumption of cloud storage is not required in the proposed scheme. In the case of the POR scheme, which is an early study, about 2% of the original data was required for verification. However, recent researches such as the proposed method generate the metadata required for verification by using the data stored in the server when the query for the verification is received. Therefore, no additional storage space is required in the server, and there is only a problem managing the key to generate verification data. The key management is not considered as a separate issue in the study.

IV. EXPERIMENTAL AND SECURITY ANALYSIS

The proposed data possession verification scheme was implemented in a cloud environment based on the Hadoop file system. For the experiment, a cloud environment was constructed by connecting twelve data nodes with the same specifications as one name node. The false positive rate of the Bloom filter for the data ownership verification index used in the proposed method was set to 0.0001. The target data were stored in a file size of 1.4 GB, and each block was set to a size of 1 MB. The purpose of the experiment was to analyze the influence of the false positive rate on the verification probability and to evaluate the temporal efficiency of the proposed method.

A. EFFICIENCY OF DATA CORRUPTION DETECTION

The data ownership verification index with the Bloom filter and false positive was used to measure how the verification probability is affected by data corruption. The corruption rate of the file stored in the server was increased in increments of 1% from 1% to 10%. For each corruption rate, the number of blocks constituting the entire file was increased in increments of 10% from 10% to 50%. The sampling process of

querying the corresponding blocks was repeated 100 times after arbitrary blocks were selected.

The execution time was measured to evaluate the effectiveness of the proposed data ownership verification technique. The storage server compared the time taken to establish the Bloom filter by accessing the block according to the block list received from the data auditor, the sampling time, and the time to obtain trapdoor. The trapdoor obtained by the auditor in response to the query from the data possession verification index was used to determine the required verification time.

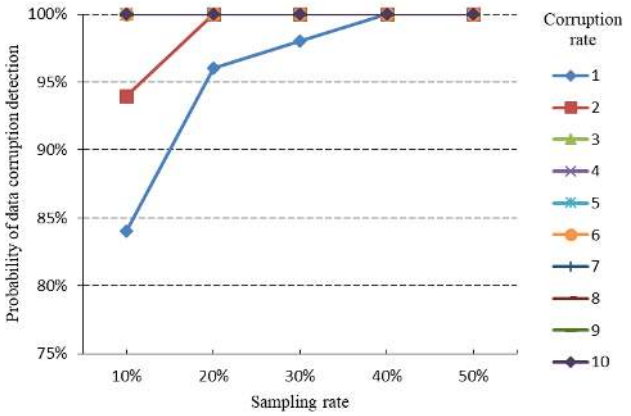

FIGURE 8. The probability of data corruption detection by sampling rate.

Fig. 8 shows the experimental results on how many blocks should be checked for data possession according to various corruption ratios of the original data. In other words, it indicates the detection rate of data corruption at different sampling rates. The x-axis represents the sampling rate, and the y-axis represents the probability that data corruption is detected. Legends 1–10 show the corruption rate of the original data.

Blocks were randomly selected from 1% to 10% for corruption. The sampling rate was increased in increments of 10% from 10% of the total block for data to 50%, while the corruption rate was increased in increments of 1%. In total, 100 validation queries were performed. The results showed that corruption was detected at a rate of 99% or more with a sampling rate of only 10% except at corruption rates of 1% and 2%.

Using the Bloom filter for verification is a similar approach to some existing techniques. However, because the proposed method generates a Bloom filter for each block, a positive error in the data of each block may not make it suitable for use. On the other hand, while the proposed scheme has a false positive rate, it is set to a small value for one file. Therefore, the false positive rate does not have a significant influence on the evaluation of PDP.

In addition, even if the corruption rate of the data is 2%, the detection rate is 99% or more at a sampling rate of 20%, and data corruption can be verified at a corruption rate is 1%. Therefore, the results in Fig. 8 show that the proposed scheme performs well compared to existing techniques even

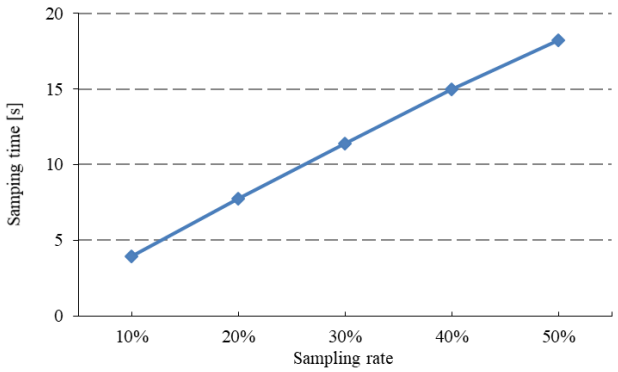

FIGURE 9. The sampling time by sampling rate.

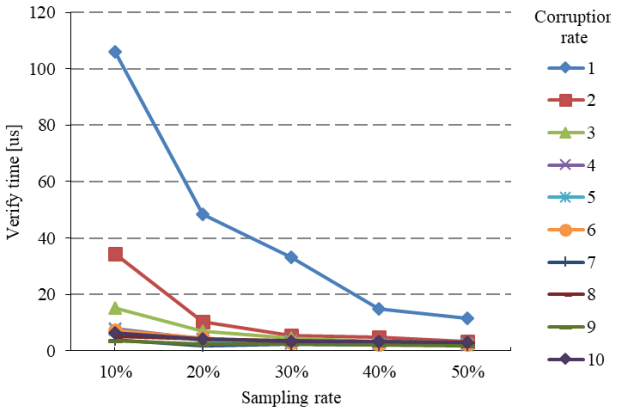

FIGURE 10. Verify time by sampling rate.

when a Bloom filter with a false positive rate is used for data possession verification.

In order to verify possession in the PDP scheme, the time required for the verification query on the server side needs to be evaluated. Fig. 9 shows the time at which the server sampled the list of blocks that received a validation query from the user. The x-axis represents the sampling rate, and the y-axis represents the sampling time. The experimental results showed that the sampling time increased linearly with the sampling rate, and the time required by human beings was the result of the second unit.

When we created a trapdoor for 10% of 1.4 GB of data, it took 4 s to read 140 MB of data with a hashing-based Bloom filter, while an RSA-based operation would take insert time. Sampling also took place on the server regardless of the environment of the user performing the verification. This means that the user did not consume many resources. Therefore, the time to verify all of the data can be verified.

Fig. 10 shows the measured time spent by the auditor to evaluate the data verification. The time required for sampling was relatively short compared to the time required for creating the trapdoor to respond to the verification request made by the server. The x-axis represents the sampling rate used to verify data possession, and the y-axis represents the time spent to verify the data possession. Legends 1–10 represent the probability of data corruption.

The experimental results showed that, because of the characteristics of the Bloom filter used to verify data possession, the detection time decreased as the sampling rate increases. This is because the probability that a corrupted block was selected increases with the sampling rate. The trapdoor value for the corruption block was effectively detected when converted to a codeword and compared with the data verification index.

The results of the detection probability of data corruption according to the sampling rate showed that the false positive of the Bloom filter did not affect the performance. Rather, when the corruption rate increased because of the characteristics of the Bloom filter, the corruption could be detected with only a small sample. In addition, although the detection time increased linearly with the sampling rate, the Bloom filter based on the hash function was evaluated to be more effective in terms of the execution speed and required resources than public key-based methods such as RSA, pairing-based cryptography, and homomorphic encryption. Therefore, if an IoT device with limited resources wants to evaluate the corruption of its own data, the proposed technique would be more effective.

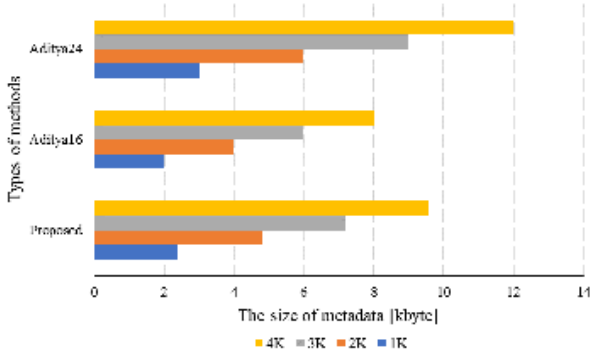

FIGURE 11. The comparison of server resource efficiency by the methods.

Recent studies have generally used dynamic provable possession when verifying data. In other words, metadata is not stored in the server but generated when queries are received from user. Nevertheless, the size of the metadata that must be generated for verification can have a significant impact on the memory of the server. Fig. 11 compares the total capacity required by the total number of blocks of the proposed method with a similar method using the Bloom filter. Aitya16 requires 16-bit metadata for one block, and Aditya24 requires 24-bit metadata. The proposed scheme uses about 14 bits per block when constructing a Bloom filter with positive error of 0.0001. As another method, SHA-1 or SHA-2 requires more than 10 times as much as 160 bits and 256 bits per block, and methods such as RSA based or HVT require the same size as the block size. Therefore, the proposed method is effective in terms of resource efficiency of the server.

B. SECURITY OF PROPOSED SCHEME

Communication between the user and the remote repository occurs to prove possession of remotely stored data. The original data should not be inferred through queries and responses in the communication. It can be assumed that the communication channel is not secure. In other words, the data that is sent and received by the user and audit system server can be assumed to be obtainable by an attacker. In this environment, the attack model of malicious attacker was introduced by J. Baek *et al.* As Game 1 and Game 2 [28]. Game 1 is a case in which the server can not be trusted, and Game 2 is an attack by an external attacker. Therefore, the audit system should be able to cope both when the server is unreliable and when it is attacked by an external attacker.

The proposed system with false positive is difficult to analyze the key used for encryption based on the collected trapdoor in the security of Game 1. It is also difficult to analyze the key of the encryption scheme by comparing the trapdoors that is generated using random keys based on known encryption scheme and is collected from public channels by attacker in the security of Game 2. Even in the proposed method, the system constituting multi-indices will increase the time to generate the trapdoor, which will be a disadvantage for the attacker [27].

V. CONCLUSION AND FUTURE WORK

Because IoT services generate massive amounts of data, a cloud computing environment is highly effective for storage and data analysis. However, because cloud storage is not owned by the user, it is untrustworthy. Therefore, the user must verify that the stored data are not corrupted. In addition, a highly effective verification method is required to handle the large amount of data. This paper proposes a secure cloud storage service based on a PDP scheme that uses a Bloom filter for an IoT environment. The proposed scheme is not based on a public key like RSA, bilinear mapping, or homomorphism. However, it can effectively prove data possession with the Bloom filter. The advantages of the proposed scheme in terms of space and speed for storing computation and metadata are competitive with those of existing techniques.

The experimental results showed that the proposed method saves time and has no significant differences in the verification rate with existing methods, even though the Bloom filter causes false positives. Therefore, the proposed service can effectively process a large amount of data generated in an IoT environment, such as smart homes. Future work will involve adjusting the level of data possession verification according to the services provided to users based on specific scenarios.

The proposed method was not applied to transferring data from a specific IoT device to cloud storage and verifying the stored data. There are many types of IoT devices, and auditing data considering the characteristics of each device takes a long time. Therefore, future work will involve analyzing the performance of the proposed scheme in a virtual machine for supporting various IoT devices and cloud storage [5].

APPENDIX

The Bloom filter is a time- and space-efficient data structure that allows for some errors. The Bloom filter is represented by an array of m bits for the set $S = s_1, \ldots, s_n$ of n elements. The bits of all arrays are initialized to zero. The filter uses r independent hash functions $h_1, \ldots, h_r$. Each element s of the set S is set to 1 in the array where $h_1\ (s), \ldots, h_r(s)$ are located. The same position may be repeatedly set to 1, but the value of a position set to 1 cannot be reset to 0. If the element a belongs to the set S, this can be determined by checking whether every position bit of $h_1(a), \ldots, h_r(a)$ is 1 or not. If all position bits identified in the array are 1, a belongs to the set S. However, Bloom filters operate based on a hash function, so false positives occur because of collisions. False positives are an error that says an element belongs to a group even though it is not actually a member. On the other hand, the Bloom filter is featured by no false negatives, where the element included in a group is not a member.

A false positive error in a search system would require another search, which would reduce the efficiency of the system. Setting the false positive rate to a small value would be sufficient. However, this requires increasing the size of the Bloom filter. A fixed-size Bloom filter changes the false positive rate depending on the number of elements stored. Therefore, a Bloom filter is needed with a false positive rate that can be effectively used in a system. If the false positive rate required by the system and the word count n of the document are determined by the characteristics of the Bloom filter, the size of the Bloom filter can be determined in m bits by the following method.

First, the false positive rate P_f required by the system is determined. P_f is the probability of a positive outcome from a search that should return a negative result. Second, the number r of keys used for the pseudo-random function from P_f is calculated. In the Bloom filter, P_f is given by (1). This rate should be minimized to satisfy (2).

$$P_f = \left(1 - (1 - 1/m)^{nr}\right)^r \approx \left(1 - e^{-rn/m}\right)^r \quad (1)$$

$$r = (\ln 2)(m/n) \quad (2)$$

$$P_f = \left(1 - e^{-\ln 2 \frac{m}{n} \times \frac{n}{m})^r}\right) = \left(1 - \frac{1}{2}\right)^r = (1/2)^r \quad (3)$$

When the system has the number of keys used in the pseudo-random function as given by (2), P_f can be calculated with (3). By converting the number of keys used in the required pseudo-random function to the expression for P_f, the following is obtained:

$$r = -\log_2 \left(P_f\right) \quad (4)$$

Finally, if the number of words contained in all the documents and the number of pseudo-random function keys are determined, the size of the Bloom filter can be calculated as follows:

$$m = nr / \ln 2 \quad (5)$$

Thus, the P_f required for a system using a Bloom filter is determined in general. This value decreases as the number of pseudo-random function keys increases. Therefore, the size of the Bloom filter is increased to reduce P_f. However, infinitely increasing the Bloom filter has an adverse effect on the search efficiency. Therefore, it is necessary to understand the characteristics of the system using the Bloom filter and to determine the parameters that optimize the Bloom filter for the system.

ACKNOWLEDGMENT

(Junho Jeong and Jong Wha J. Joo contributed equally to this work.)

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] W. Joe, M. Jiang, and K. Jeong, "An M2M/IoT based smart data logger for environmental sensor networks," *J. KIISE, Comput. Practices Lett.*, vol. 20, no. 1, pp. 1–5, 2014.

[4] S. Kim, J.-H. Kim, J. Yun, and S. H. Lee, "Machine learning-based temperature control for smart home environment," in *Proc. EEECS*, Dubrovnik, Croatia, 2017, pp. 35–39.

[5] Y. Lee, J. Jeong, and Y. Son, "Design and implementation of the secure compiler and virtual machine for developing secure IoT services," *Future Gener. Comput. Syst.*, vol. 76, pp. 350–357, Nov. 2017.

[6] P. Wang, D. J. Dean, and X. Gu, "Understanding real world data corruptions in cloud systems," in *Proc. IEEE Int. Conf. Cloud Eng.*, Tempe, AZ, USA, Mar. 2015, pp. 116–125.

[7] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[8] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, "Dark clouds on the horizon: Using cloud storage as attack vector and online slack space," in *Proc. 20th USENIX Conf. Secur.*, San Francisco, CA, USA, 2011, p. 5.

[9] T. Shuang, C. Zhi-Kun, and Z. Jian-Feng, "Data blocks' signature in cloud computing," in *Proc. Int. Symp. Comput. Bus. Intell.*, New Delhi, India, Aug. 2013, pp. 49–55.

[10] T. Shuang, T. Lin, L. Xiaoling, and J. Yan, "An efficient method for checking the integrity of data in the cloud," *China Commun.*, vol. 11, no. 9, pp. 68–81, Sep. 2014.

[11] J. Li, M. N. Krohn, D. Mazieres, and D. E. Shasha, "Secure untrusted data repository (SUNDR)," in *Proc. OSDI*, San Francisco, CA, USA, 2004, pp. 121–136.

[12] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. CCS*, Alexandria, VA, USA, 2007, pp. 598–609.

[13] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw.*, 2008, Art. no. 9.

[14] C. C. Erway, A. Küpçü, C. Papamanthou, and B. Tamassia, "Dynamic provable data possession," in *Proc. CCS*, Chicago, IL, USA, 2009, pp. 213–222.

[15] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in *Proc. CCS*, Alexandria, VA, USA, 2007, pp. 584–597.

[16] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. CCSW*, Chicago, IL, USA, 2009, pp. 43–54.

[17] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2008, pp. 90–107.

[18] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.

[19] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. 17th Int. Workshop Qual. Service*, Charleston, SC, USA, Jul. 2009, pp. 1–9.

[20] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. ESORICS*, Saint Malo, France, 2009, pp. 355–370.
[21] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage," in *Proc. CCS*, Chicago, IL, USA, 2009, pp. 187–198.
[22] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, Dec. 2012.
[23] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.
[24] M. Liu, X. Wang, C. Yang, Z. L. Jiang, and Y. Li, "An efficient secure Internet of Things data storage auditing protocol with adjustable parameter in cloud computing," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 1, pp. 1–11, 2017.
[25] W. Sai, X. Zhang, C. Xie, H. Li, and H. Zhang, "The application of cloud data integrity verification scheme in Internet of Things security," in *Proc. 14th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process.*, Sichuan, China, Dec. 2017, pp. 268–273.
[26] T. Aditya, P. K. Baruah, and R. Mukkamla, "Space-efficient bloom filters for enforcing integrity of outsourced data in cloud environments," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, Washington, DC, USA, Jul. 2011, pp. 292–299.
[27] J. Jeong and Y. S. Hong, "Efficient multi-indices scheme for searchable encryption system against brute force attack in cloud computing environments," *J. KISS, Inf. Netw.*, vol. 40, no. 5, pp. 286–293, 2013.
[28] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proc. Int. Conf. Comput. Sci. Appl.*, Perugia, Italy, 2008, pp. 1249–1259.

JUNHO JEONG received the B.S. degree from the Department of Computer Science and Engineering, Dongguk University, Seoul, South Korea, in 2007, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, Dongguk University, Seoul, in 2009 and 2015, respectively, where he is currently a Research Professor with the Department of Computer Science and Engineering. His research areas include computer security, privacy-preserving, distributed systems, network security, and secure software.

JONG WHA J. JOO received the B.S. degree in computer science and engineering from Seoul National University, Seoul, South Korea, in 2005, the M.S. degree in computer science from Brown University, Providence, RI, USA, in 2007, and the Ph.D. degree in bioinformatics from the University of California, Los Angeles, CA, USA, in 2016. She is currently an Assistant Professor with the Department of Computer Science and Engineering, Dongguk University, Seoul. Her research interests include developing efficient computational methodologies and algorithms for genome-wide association studies and expression quantitative trait loci studies.

YANGSUN LEE received the B.S. degree from the Department of Computer Science, Dongguk University, Seoul, South Korea, in 1985, and the M.S. and Ph.D. degrees from Department of Computer Engineering, Dongguk University, in 1987 and 2003, respectively. He was the Manager of the Computer Center, Seokyeong University, from 1996 to 2000, the Director of the Korea Multimedia Society, from 2004 to 2018, the General Director of the Korea Multimedia Society, from 2005 to 2006, the Vice President of the Korea Multimedia Society, in 2009, and the Senior Vice President of the Korea Multimedia Society, in 2015. Also, he was the Director of the Korea Information Processing Society, from 2006 to 2014, and the President of the Society for the Study of Game at Korea Information Processing Society, from 2006 to 2010. Moreover, he was the Director of HSST, from 2014 to 2018. He is currently a Professor with the Department of Computer Engineering, Seokyeong University, Seoul. His research areas include smart system solutions, programming languages, and embedded systems.

YUNSIK SON received the B.S. degree from the Department of Computer Science and Engineering, Dongguk University, Seoul, South Korea, in 2004, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, Dongguk University, in 2006 and 2009, respectively. He was a Research Professor with the Department of Brain and Cognitive Engineering, Korea University, Seoul, South Korea, from 2015 to 2016. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Dongguk University. Also, his research areas include secure software, programming languages, compiler construction, and mobile/embedded systems.