A Succinct Provable Data Possession Mechanism for Lightweight Clients in Network Computing

Yang Xu, Quanrun Zeng, Ju Ren, Yaoxue Zhang
School of Information Science & Engineering
Central South University
Changsha, China
{xuyangcsu, zengqquanrun, renju, zyx}@csu.edu.cn

Guojun Wang
School of Computer Science & Technology
Guangzhou University
Guangzhou, China
csgjwang@gmail.com

Hao Min
School of Software
Central South University
Changsha, China
miao95@csu.edu.cn

***Abstract*—The wide use of network computing technologies has promoted the popularity of outsourced storage services. Meanwhile, the provable data possession (PDP) techniques are widely studied as the outsourcing storage servers are not always trustworthy and dependable in maintaining the outsourced data. However, most existing PDP solutions are either costly for the current Internet of things (IoT) devices, or vulnerable to spoofing attacks. In this paper, we propose a succinct anti-spoofing provable data possession scheme for lightweight clients in network computing. We employ the basic integrity comparison-based verification to fit for the resource-constrained verifiers, and propose a random sentinel metadata setup mechanism and the true-fake blended challenge scheme to improve the robustness against spoofing attacks of untrusted servers. The analyses reveal that our approach is effective in data possession verification with the robustness against spoofing attacks, while the overhead on the verifier side is low.**

***Keywords*—Network Computing, Outsourced Storage, Provable Data Possession, Anti-spoofing, Lightweight Client**

I. Introduction

In the era of network computing, new computing technologies such as cloud computing [1], transparent computing [2], [3] and edge computing [4], [5] provide terminal devices with all-embracing services via the network. In this context, storage-outsourcing services which enable terminals to store huge amount of data in remote storage servers for reducing the burden of local data maintenance are generally adopted and extensively used in practice [6].

However, there are also some disadvantages to the storage-outsourcing pattern. The outsourcing storage systems are not always trusted enough to keep the outsourced data while the clients no longer physically possess their data. These storage providers may lose data accidentally due to hardware failures, or even worse, delete the rarely used data deliberately for their own interests.

For mitigating this problem, the provable data possession (PDP) techniques which allow clients to verify whether their data are still stored in untrusted outsourcing servers has been widely studied academically and industrially [7]. Since the simple integrity-based outsourcing data verification technique has been proposed by Deswarte et al. [8] in the early days, numerous improvements have been made progressively in terms of attack resistance [9], [10], dynamic modification [11], [12], multi-copy support [13], public verifiability [14], retrievability [15], [16], etc. Unfortunately, existing solutions cannot resolve the problem entirely in current practical scenarios. With the flourish of the Internet of things (IoT) industry [17], numerous resource-constrained IoT devices generate and outsource massive amounts of data to remote storage servers continuously. For instance, wearable smart-bands are keeping producing and updating the activity data of the owners to remote storage servers every day. Consequently, for one thing, the high cryptographic computational overheads of the verification procedures of most PDP solutions are not affordable to all the resource-limited IoT devices. For another, few lightweight approaches are resistant to the spoofing attacks from untrusted servers, especially with the enhancements of statistic analysis abilities of the adversaries to infer and reserve the corresponding verification metadata with the aid of current big data and deep learning techniques [18].

In this paper, we propose a succinct anti-spoofing provable data possession scheme for lightweight clients in network computing from a novel perspective. We employ the typical challenge/response verification model and adopt the succinct verification method in which the verifier simply compares the reply hash checksums of data blocks with the pre-reversed sentinel ones, to minimize the overhead on the resource-constrained verifier side. Furthermore, to make the efficient approach robust to spoofing attacks from untrusted servers, we not only pre-reverse hash checksums of sequential data blocks with fixed size and random adjacent overlaps as unpredictable sentinel metadata to protect against guessing attacks, but also conceal every true verification intent (associated with really possessed sentinel metadata) with a sufficient amount of indistinguishable spurious ones during the verification to defend against statistical replay attacks. Finally, we discuss our approach in terms of security and performance, which shows that it is effective in data possession verification with the robustness against spoofing attacks and low overhead on the verifier side.

In summary, our major contributions are listed as follows:

1. We pre-reverse the position indices and hash checksums of selected sentinel data blocks which overlap each other randomly and cover the whole outsourcing data as the sentinel metadata for the challenge, to prevent the untrusted servers from guessing these sentinel blocks associated with the sen-

tinel metadata possessed by the verifier.

2. We obscure the true verification intent involving sentinel metadata with the sufficient and statistically indistinguishable fake ones to defend against spoofing attacks as the untrusted servers can neither recognize the true sentinel data blocks even through statistical analyses nor be benefited from keeping every possible sentinel metadata instead of real data to respond all possible challenges.

3. We prove the security of our approach against spoofing attacks from untrusted servers and theoretically analyze the performance of our proposed approach.

The remainder of this paper is organized as follows. Section II summarizes the related work. Section III describes the threat model. In Section IV, we propose a novel succinct PDP scheme suitable for lightweight clients. And then, in Section V, we analyze the effectiveness and performance of our approach. Finally, in the last section, we conclude our work and propose the possible improvements in the future.

II. Related Work

Since the outsourcing data services are susceptible to accidental or malicious data corruption, the PDP techniques, which enable the clients to verify their outsourcing data stored in untrusted remote servers based on data integrity, have naturally become the research hotspot.

Deswarte et al. [8] designed and implemented a basic and crude solution for outsourcing data verification in the early days. In their approach, the verifier just pre-computes and stores several excepted hash checksums of the data concatenated with auxiliary secret numbers before data outsourcing, so that the verifier can challenge the server with the secret numbers and then compares the responses with the pre-computed ones for data possession verification. However, such a simple PDP solution is vulnerable to spoofing attacks as the verification metadata (i.e., secret numbers and corresponding hash values) possessed by the verifier are effective only in one-time because a malicious server can keep the hash values after challenges for further use without maintaining corresponding data.

Ateniese et al. [9] formalized a typical static PDP model (S-PDP). They exploited the homomorphic hash technique to implement the repeatable integrity verification for outsourcing data and employed a probabilistic data verification with the help of the random sampling technique to cut down the communication cost. Their results showed that the approach was able to detect the imperceptible server misbehavior (1% data corruption) with high probability ($\geq 99\%$) through only 4.6% data verification.

In order to support the dynamic data modification, Ateniese et al. [11] further proposed a scalable PDP scheme (ScPDP) with finite-repeatable data appending capability. However, this approach has shortcomings in effectiveness as it only enables a finite number of valid verification and is somewhat vulnerable to guessing attacks. Erway et al. [12] proposed a series of dynamic PDP (DPDP) models relying on the rank-based skip list, which supports multiple updating operations on the provable data. But in these schemes, the computing overhead on the client sides still need to be reckoned substantially.

Barsoum et al. [13] proposed a new PDP technique which enables slight modifications on multiple data copies so as to support the multi-copy storage scenarios. To a certain extent, this solution is also capable to mitigate prediction attacks. However, such a technique brings about non-negligible overhead on the client sides, which is unacceptable for resource-constrained verifiers.

Vijaya Kmari et al. [14] introduced a PDP approach which enables probabilistic proofs of possession by sampling random sets of data blocks from the server in order to reduce I/O costs. A trusted third party (TTP) is also introduced to share the computing overhead on clients. Unfortunately, the TTP is not always available in many practical environments.

Besides, by adding redundant recovery information into the metadata possessed by the verifier, the proof of retrievability (POR) scheme is implemented on the basis of data integrity verification [15], [16], which additionally enables data recovery when the verification results turn out to be negative. However, the POR technique is severely hindered by its limited number of valid verification and recovery, as well as the significant overhead.

When it comes to lightweight clients, e.g., the massive IoT devices, the cryptographic computational overheads of most existing solutions are unaffordable to many verifier devices due to the resource limitations, while few lightweight schemes are resistant to the spoofing attacks, especially considering the improvements of adversaries by leveraging current big data and artificial intelligence techniques. Therefore, an efficient and secure PDP approach is still needed by lightweight clients in practical network computing environments.

III. Threat model

We assume that the untrusted server may spoof clients by maintaining the small-size metadata instead of the large original data for its own interests (e.g., saving storage costs or attempting to conceal the facts of data loss). It is able to record all the challenge information for analyses in order to generate the right responses for cheating the verifier during the verification. However, the adversary cannot defeat current cryptographic technique like standard hash function directly.

IV. Approach

In this section, we first describe the general idea of our approach and then detail it in terms of the setup phase and verification phase respectively.

A. Overview

U nlike most existing PDP solutions relying heavily on high-cost cryptographic computation on both client and server sides for verification, our approach achieves this goal from the obfuscating point of view to satisfy the lightweight verification clients. Our main idea is that, before outsourcing the data to an untrusted storage server, the verifier simply pre-computes and stores several hash checksums of sentinel data blocks which overlap each other randomly and cover the whole data, as the expected responses of challenges.

When verification is required, the verifier challenges the storage server with only a set of block-position indices consisting of a true one (i.e., the one associated with existing sentinel metadata) accompanied by several obfuscating fake ones (no sentinel metadata is pre-reserved) for security purpose. For the response, storage server should compute corresponding hash checksums of all these specified data blocks according to the position indices and then reply them to the verifier. And the verifier just ignores the invalid replies and simply compares the true one with the pre-reversed sentinel one for proof, which is extremely efficient for the verifier in term of computation of verification.

B. Setup phase

In the setup phase, as for the data D which is going to be outsourced, the verifier will generate and store a set of sentinel metadata in advance for future challenge and verification.

A piece of sentinel metadata is a pair of index and hash checksum of corresponding fixed-size sentinel data block belonging to the data D. Here, the block position is used as the block index, e.g., the pos is the index of the data block $B[pos][pos + bs]$, where bs denotes the block size[1].

The verifier starts at a random position of D (using the step size ss as the skip granularity, where $ss \in [1\ bit, hs]$ and hs denotes the size of hash checksum[2] of a data block) and selects sentinel data blocks sequentially. The adjacent blocks will overlap each other by random steps, so as to make the position indices of these true sentinel metadata uncorrelated and unpredictable, thereby further preventing the untrusted servers from guessing these data blocks for preparing related checksums (see details in Section V). Besides, to ensure the verifiability of the entire data, the verifier keeps selecting data blocks in sequence for hash operations until covering the whole data. Note that as for the last block of D, it will be fulfilled by the bits from the start of D if there are not enough bits left in the end.

As for each selected data block, the verifier simply applies the hash function to produce the corresponding hash checksum, which is the expected response from the server during the verification, i.e., $M_{pos} = Hash(B[pos][pos + bs])$ is the right answer of the challenged block index pos. Consequently, the storage cost of sentinel metadata is limited due to the small size of the hash output. The details are given in Algorithm 1.

Additionally, the verifier divides the entire data D into several partitions with the partition size of no less than $(\lceil \frac{bs}{hs} \rceil \times ss)$ and keeps the auxiliary partition information. Therefore, each partition is able to accommodate no less than $\lceil \frac{bs}{hs} \rceil$ block indices theoretically and each selected block position index falls within a certain partition.

[1]In practice, the block size is usually set as several Mb with the upper bound of $D.size - 1$.

[2]Practically, the standard hash functions such as SHA-1 (128 bits output), SHA-256 (256 bits output), SHA-512 (512 bits output), etc., are commonly used.

Algorithm 1 Sentinel metadata generating algorithm

```
Input: the data $D$, the block size $bs$ and the step size $ss$
Output: the sentinel metadata $smd$
  Select a random position $pos$ as the start position
  while $data$ is not totally covered do
    $pos \leftarrow pos + ((\text{Random}(i) \times ss) \mod bs)$
    #Picking up a sentinel data block and calculating its hash
    checksum
    if $(pos + bs) \leq D.size$ then
      $M \leftarrow \text{Hash}(B[pos][pos + bs])$
    else
      $M \leftarrow \text{Hash}(B[pos][D.size] + B[0][pos + bs - D.size])$
    end if
    $smd.push(< pos; M >)$
  end while
  return $smd$
```

Finally, the data D, as well as the block size bs, will be outsourced to the remote storage server.

C. Verification Phase

In order to verify the possession of a sentinel data block $B[i][i + bs]$ stored on the remote storage server, the verifier challenges the server with the position index of the certain partition within which the corresponding sentinel block index pos_i (i.e., the one associated with the really possessed sentinel metadata pair $< pos_i; M_i >$) as well as other $\lceil \frac{bs}{hs} \rceil - 1$ indistinguishable fake ones fall[3] (step 1 in Fig. 1), instead of submitting the single block index of $B[i][i + bs]$, so as to prevent the untrusted server from recognizing the true sentinel data blocks directly or with the help of statistical analyses, or being benefited from keeping every possible hash checksums instead of original data for successfully responding all possible challenges (see details in Section V).

When received the challenge from the verifier, the server would figure out the true-fake blended block indices included in the certain partition based on the position index of the partition together with the step size, and then computes the corresponding hash checksums of all the specified data blocks according to these indices equally (step 2 in Fig. 1), e.g., $M'_a = Hash(B[a][a+bs]), \ldots, M'_m = Hash(B[m][m+bs])$, and replies the set of $< pos_*; M'_{pos_*} >$ pairs to the verifier (step 3 in Fig. 1).

As for the checksums returned by the server, the verifier just ignores the ones associated with the fake block indices and only checks whether the one related to the true index is equal to that in the stored sentinel metadata (step 4 in Fig. 1). If the check succeeds, the verifier believes that the storage server is still possessing that data block[4].

Apparently, our scheme incurs truly minimal overhead on the client side during the verification phase as no high-cost

[3]Although there may exist other true block indices in the certain partition, these true ones are treated as the fake ones as well in a certain verification.

[4]According to the strand PDP [9], the verifier is able to discover the server misbehaviors with a high probability through the successful verifications of only a small portion of all the data blocks.

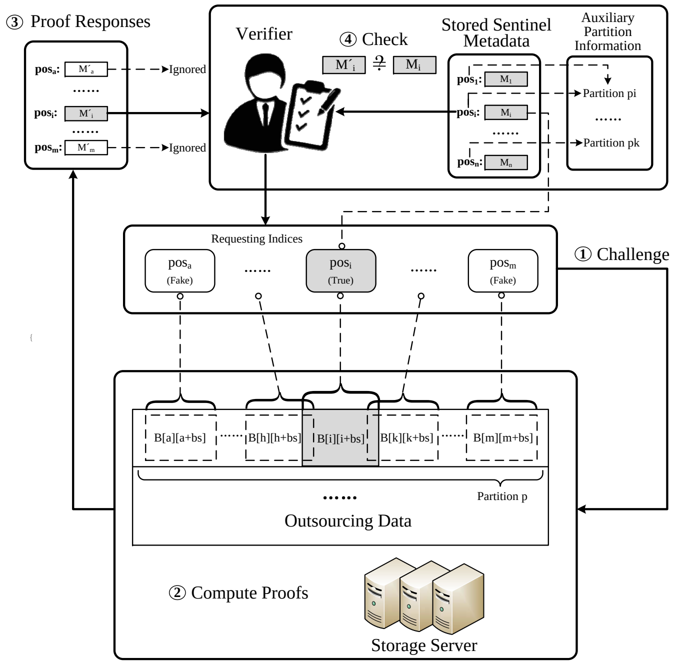

Fig. 1. The True-fake Blended Verification Process

cryptographic operations are needed in order to check the reply from the server. Moreover, it is also obvious that our scheme allows unlimited verifications and is publicly verifiable if outsourcing the sentinel metadata to any third party.

V. Discussion

In this section, we discuss the security, effectiveness and performance of our approach.

A. Security & Effectiveness

Being different from most PDP solutions which rely on cryptographic mechanisms to ensure the security, our approach leverages the obfuscating method instead to achieve the security verification.

Although we employ no additional cryptographic protection mechanism such as challenge nonce for defending the vulnerable sentinel metadata, we conceal the true sentinel block index with at least $\lceil \frac{bs}{hs} \rceil - 1$ indistinguishable fake ones from the same partition. Therefore, it is still almost impossible for the untrusted server to pick out the right index of the sentinel data block from a large amount of the confusing fake ones in a verification.

Specifically, the success rate of guessing attack for the malicious server during a verification is less than $\frac{1}{\lceil \frac{bs}{hs} \rceil}$. Since bs is usually millions of times larger than hs, this rate approaches to 0 in practice.

As the size of a hash checksum is much smaller than that of a data block, there is no doubt that the untrusted server has the cheating motivation when it is capable of providing a forged possession proof of a certain sentinel block by maintaining all the hash checksums (a true one is definitely included) of that certain block verification with low storage cost.

Fortunately, in our approach, a true block index is mixed with at least $\lceil \frac{bs}{hs} \rceil - 1$ indistinguishable fake ones in each verification. Hence, it remains unprofitable for the malicious server as it has to store no less than $\lceil \frac{bs}{hs} \rceil$ hash checksums (the total size is no less than $(\lceil \frac{bs}{hs} \rceil \times hs)$ and is no less than that of an original block) to forge the possession proof of a certain sentinel block. Therefore, the storage server has no reason to cheat because the cheating cost is as higher as that of being honest.

Furthermore, as the malicious server might try to distinguish the sentinel blocks from the fake ones according to the statistical frequency analyses of the block indices, to address this potential threat, whenever a certain sentinel block is going to be verified, the server will also be required to indiscriminately provide the possession proofs of all the other possible blocks associated with the other obfuscating indices falling within the same partition at the same time, which ensure that the querying frequency of the obfuscating fake indices are basically indistinguishable from that of the true ones statistically.

Besides, since we select the sentinel data blocks with random adjacent overlaps, there is neither predictable pattern in how those sentinel data blocks are distributed in the entire outsourcing data D, nor clear correlations among the sentinel data blocks, in other words, the server will never have additional advantages for recognizing more sentinel data blocks even if one sentinel block is disclosed.

Additionally, according to the security analyses above, it is obvious that there exist no consumptive factors which limit the number of verifications, therefore, our scheme is capable of supporting infinite secure verifications.

B. Performance

As we mentioned above, the high computational overhead in most existing PDP schemes is disadvantageous to the PDP implementation for the lightweight clients such as IoT devices in the network computing environment. By replacing the cryptographic computation involved in most PDP schemes with the confusion process, our approach achieves the unlimited repeatable verification with light computing overhead on the client side. Being different from the DPDP and some other existing methods, our clients only need to check whether the checksum returned by the server is equal to the pre-stored one, while in most PDP schemes (such as DPDP [12] and SPDP [11]) the decryption calculation is needed. Without decryption or other complex calculations, the computing overhead on the client side would be extremely low in our approach. When it comes to the server side, a server needs to compute the hash checksum of the sentinel block as well as the fake ones. For each really possessed sentinel block index, the server will have to compute at least $(\lceil \frac{bs}{hs} \rceil \times ss)$ hash checksums. That leads to an additional computing overhead on the server.

As for storage overhead, the client needs to record around $\lceil \frac{D.size}{bs} \rceil$ sentinel block indices, along with the corresponding hash checksums and the auxiliary partition information (which can be considered as a small constant). Therefore, the storage

TABLE I
PERFORMANCE COMPARISON

Performance Indicator \ Scheme	RIC [8]	POR [15]	S-PDP [9]	ScPDP [11]	DPDP [12]	Our Approach
Repeatable verification	no	Limited	yes	yes	yes	yes
Computing overhead (Client side)	low	medium	medium	medium	medium	low
Storage overhead (Client side)	low	low	low	low	low	low
Communication overhead (Client side)	medium	medium	low	low	medium	low
Computing overhead (Server side)	medium	medium	medium	medium	medium	high
Storing overhead (Server side)	low	high	low	low	high	low
Communication overhead (Server side)	medium	medium	low	low	medium	high

space required for the client is approximately $\lceil \frac{D.size}{bs} \rceil \times hs$. As the block size bs is usually larger than the size of the hash value hs significantly, the storage overhead on the client side is low. In addition, the server does not need to keep any extra information except the stored data, i.e., the storage overhead is very small from the entire perspective.

When considering the communication overhead, for each request initiated, the only information that the client needs to send is the partition index, therefore the transmission overhead on the client side is very small, nearly the same as most schemes which require sending indices. Similar to the computing cost, in our approach, the communication overhead on the server side will be slightly higher than most PDP schemes, because the server not only needs to read a lot of data to calculate the hash checksums, but also has to transmit many fake requests. But on the whole, considering the fact that the hash output has a reasonable size, and the client can simply ignore the hash checksums of fake block indices, the increase of communication overhead is acceptable.

Consequently, our scheme incurs lower overhead on the client side when compared with other schemes. More detailed comparisons are shown in Table 1.

VI. Conclusion

In this paper, we propose a provable data possession scheme for lightweight clients in network computing with the spoofing resistant feature. We simply adopt the general integrity comparison-based verification that incurs very little overhead of the resource-limited verifiers. Meanwhile, for mitigating the security vulnerabilities associated with the simple verification method, we not only propose a random sentinel metadata setup mechanism but also the true-fake blended challenge scheme to make the approach more robust to spoofing attacks. The analyses show that our approach is effective in verifying the possession of own data stored in untrusted servers and is robust against spoofing attacks, while the overhead on the verifier side is low.

As for the further enhancements, we would like to adopt the variable-size block pattern in the sentinel metadata setup procedure so as to further improve the diversity and randomness of metadata possessed by the verifiers. Furthermore, we plan to implement and experimentally evaluate our approach on our medical big data system consisting of a big data platform and multitudinous health monitoring IoT devices.

Acknowledgment

This research was supported in part by the National Natural Science Foundation of China under Grants: 61632009, 61702562 and 61472451, the Hunan Provincial Innovation Foundation for Postgraduate under the Grant: CX2015B047, the Guangdong Provincial Natural Science Foundation under the Grant: 2017A030308006, High-Level Talents Program of Higher Education in Guangdong Province under Grant 2016ZJ01 and the Joint Research Project between Tencent and Central South University.

References

[1] Mell, Peter M., and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards & Technology, 2011.
[2] J. He, Y. Zhang, J. Lu, M. Wu, and F. Huang, "Block-stream as a Service: A More Secure, Nimble, and Dynamically Balanced Cloud Service Model for Ambient Computing," IEEE Network, vol. 32, pp. 126–132, 2018.
[3] Y. Zhang, K. Guo, J. Ren, Y. Zhou, J. Wang, and J. Chen, "Transparent Computing: A Promising Network Computing Paradigm," Computing in Science & Engineering, vol. 19, pp. 7–20, 2017.
[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, pp. 637–646, 2016.
[5] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing," IEEE Network, vol. 31, pp. 96–105, 2017.
[6] International Data Corporation (IDC), "Storage and Data Management Services," Technical Report, 2017.
[7] A. Barsoum, "Provable Data Possession in Single Cloud Server: A Survey, Classification and Comparative Study," International Journal of Computer Applications, vol. 123, No. 9, pp. 1–10, 2015.
[8] Y. Deswarte, J. J. Quisquater, and A. Sadane, "Remote Integrity Checking," Integrity and Internal Control in Information Systems VI, pp. 1–11, 2004.
[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," in Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), pp. 598–609, 2007.
[10] G. Xu, Y. Yang, C. Yan, and Y. Gan, "A Probabilistic Verification Algorithm Against Spoofing Attacks on Remote Data Storage," International Journal of High Performance Computing & Networking, vol. 9, No. 3, pp. 218–229, 2016.
[11] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," in Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm), pp. 1–10, 2008.
[12] C. C. Erway, A. Kpç, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," in Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), pp. 213–222, 2009.
[13] A. F. Barsoum and M. A. Hasan, "Provable Multicopy Dynamic Data Possession in Cloud Computing Systems," IEEE Transactions on Information Forensics and Security, vol. 10, pp. 485–497, 2015.

[14] P. Vijaya Kmari, V. Kavitha and Naresh A., “Cloud Storage Auditing by Utilizing Provable Data Possession Method,” in Proceedings of 2018 IADS International Conference on Computing, Communications & Data Engineering (CCODE), pp. 1–5, 2018.

[15] K. D. Bowers, A. Juels, and A. Oprea, “HAIL:A High-availability and Integrity Layer for Cloud Storage,” in Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS), pp. 187–198, 2009.

[16] H. Shacham, B. Waters, “Compact Proofs of Retrievability,” Journal of Cryptology, vol. 26, No. 3, pp. 442-483, 2013.

[17] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions,” Future Generation Computer Systems, vol. 29, No. 7, pp. 1645–1660, 2013.

[18] J. Soria-Comas, J. Domingo-Ferrer, “Big Data Privacy: Challenges to Privacy Principles and Models,” Data Science & Engineering, vol. 1, No. 1, pp. 21–28, 2016.