International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

Dynamic and Public Auditing With Fair Arbitration

S.FARHANA U.SANDHYA

Abstract—Cloud users no longer physically possess their data, so how to ensure the integrity of their outsourced data becomes a challenging task. Recently proposed schemes such as "provable data possession" and "proofs of retrievability" are designed to address this problem, but they are designed to audit static archive data and therefore lack of data dynamics support. Moreover, threat models in these schemes usually assume an honest data owner and focus on detecting a dishonest cloud service provider despite the fact that clients may also misbehave. This paper proposes a public auditing scheme with data dynamics support and fairness arbitration of potential disputes. In particular, we design an index switcher to eliminate the limitation of index usage in tag computation in current schemes and achieve efficient handling of data dynamics. To address the fairness problem so that no party can misbehave without being detected, we further extend existing threat models and adopt signature exchange idea to design fair arbitration protocols, so that any possible dispute can be fairly settled. The security analysis shows our scheme is provably secure, and the performance evaluation demonstrates the overhead of data dynamics and dispute arbitration are reasonable.

1 INTRODUCTION

DATA oursourcing is a key application of cloud computing, which relieves cloud users of the heavy burden of data management and infrastructure maintenance, and provides fast data access independent of physical locations. However, outsourcing data to the cloud brings about many new security threats. Firstly, despite the powerful machines and strong security mechanisms provided by cloud service providers (CSP), remote data still face network attacks, hardware failures and administrative errors. Secondly, CSP may reclaim storage of rarely or never accessed data, or even hide data loss accidents for reputation reasons. As users no longer physically possess their data and consequently lose direct control over the data, direct employment of traditional cryptographic primitives like hash or encryption to ensure remote data's integrity may lead to many security loopholes.In particular, downloading all the data to check its integrity is not viable due to the expensive communication overhead,especially for large-size data files.

In this sense, message authentication code (MAC) or signature based mechanisms, while widely used in secure storage systems, are not suitable for integrity check of outsourced data, because they can only verify the integrity of retrieved data and do not work for

rarely accessed data (e.g., archive data). So how to ensure the correctness of outsourced data without possessing the original data becomes a challenging task in cloud computing, which, if not effectively handled, will impede the wide deployment of cloud services. Data auditing schemes can enable cloud users to check the integrity of their remotely stored data without down-. loading them locally, which is termed as blockless verification. With auditing schemes, users can periodically interact with the CSP through auditing protocols to check the correctness of their outsourced data by verifying the integrity proof computed by the CSP, which offers stronger confidence in data security because user's own conclusion that data is intact is much more convincing than that from service providers. Generally speaking, there are several trends in the development of auditing schemes. First of all, earlier auditing schemes usually require the CSP to generate a deterministic proof by accessing the whole data file to perform integrity check, e.g., schemes in use the entire file to perform modular exponentiations. Such plain solutions incur expensive computation overhead at the server side, hence they lack efficiency and practicality when dealing with large-size data. Represented by the "sampling" method in "Proofs of Retrievability" (PoR) [3] model and "Provable Data Possession" (PDP) [4] model, later schemes tend to provide a probabilistic proof by accessing part of the file, which obviously enhances the auditing efficiencyover earlier schemes.

2 **RELATED WORK**

Remote integrity check could be sourced to memory check schemes that aim to verify read and write operations to a remote memory. Recently, many auditing schemes have been proposed around checking the integrity of outsourced data. Deswarte et al. [1] and Filho et al. [2] use RSA-based hash functions to check a file's integrity. Although their approaches allow unlimited auditing times and offer constant communication complexity, their computation overhead is too expensive because their schemes have to treat the whole file as an exponent. Opera et al. propose a scheme based on tweakable block

cipher to detect unauthorized modification of data blocks, but verification needs to retrieve the entire file, thus the overhead of data file access and communication are linear with the file size. Schwarz et al. propose an algebraic signature based scheme, which has the property that the signature of the parity block equals to the parity of the signatures on the data blocks. However, the security of their scheme is not proved. Sebe et al. provide an integrity checking scheme based on the Diffie-Hellman problem. They fragment the data file into blocks of the same size and fingerprint each data block with an RSAbased hash function. But the scheme only works when the block size is much larger than the RSA modulus *N*, and it still needs to access the whole data file. Shah et al. propose a privacy-preserving auditing protocol that allows a third party auditor to verify the integrity of remotely stored data and assist to extract the original data to the user. As their scheme need firstly encrypt the data and precompute a number of hashes, the number of auditing times is limited and it only works on encrypted data. Furthermore, when these hash values are used up, the auditor has to regenerate a list of new hash values, which leads to extremely high communication overhead. From above analysis, it can be seen that earlier schemes usually generate a deterministic proof by accessing the whole data file, thus their efficiency is limited due to the high computation overhead. To address this problem, later schemes tend to generate a probabilistic proof by accessing part of the date file. Jules et al. propose a proofs of retrievability (PoR) model, where spot-checking and errorcorrecting code are used to guarantee the possession and retrievability of remote stored data. However, PoR can only be applied to encrypted data, and the number of auditing times is a fixed priori due to the fact that sentinels embedded in the encrypted data could not be reused once revealed. Dodis el al. identify several other variants of PoR in .Ateniese et al. [4] are the first to put forward the notion of public verifiability in their provable data possession (PDP) scheme, where the auditing tasks can be delegated to a third-party auditor. In PDP, they propose to randomly sample a few data blocks to obtain a probabilistic proof, which greatly reduces the computation overhead. Moreover, PDP scheme allows unlimited number of auditing. Shacham et al. [5] design an improved PoR scheme and

provide strict security proofs in the security model defined in [3], they use homomorphic authenticators and provable secure BLS signatures to achieve public verifiability, which is not provided in Jules' main PoR scheme. Some other schemes with public auditability aim to provide privacy protection against information leakage toward a third-party auditor in the process of integrity auditing. However, all above-mentioned schemes are designed for static data only, direct extension of these schemes to support data dynamics may suffer from security problems, as analyzed in [14]. But in cloud environment, remotely stored data may not only be read but also be updated by users, which is a common requirement. In this sense, schemes can only audit static data is insufficient and lacks of practicability. To support data dynamics in auditing schemes, Ateniese et al. propose a dynamic version of their original PDP scheme using symmetric encryption, however, the number of auditing times is limited and fully block insertion is not supported (only append-type insertion is supported).Erway et al. [9] firstly propose to construct a fully dynamic provable data possession (DPDP) scheme. To eliminate the index limitation of tag computation in original PDP scheme and avoid tag re-computation brought by data dynamics, they use the rank of a skip list node (similar to block index) to uniquely differentiate among blocks and authenticate the tag information of challenged blocks before proof verification. However, the skip list in essence is an authenticated structure used to test set-membership for a set of elements. To prove the membership of a specific node, a verification path from the start node to the queried node must be provided, its communication cost is linear to the number of challenged blocks. Moreover, there's no explicit implementation of public verifiability given for their scheme. Qian Wang et al. [6] combine BLS signature based homomorphic authenticator with Merkle hash tree to provide both public auditability and fully dynamic operations support. Specifically, their scheme constructs a Merkle hash tree, stores the hashes of tags in the leaf nodes and recursively computes the root and signs it, which is used to authenticate the tags of challenged blocks. Furthermore, they eliminate the index limitation in tag computation by using $H(mi)$ to replace $H(name||i)$ in [5], which requires blocks to be different with each other. However, such a requirement on data blocks is not

appropriate since the probability of block resemblance increases when block size decreases. In addition, due to the authentication of challenged blocks with a Merkle Hash Tree, the communication cost of their scheme is also linear to the number of requested blocks. Zhu et al. [10] use index-hash table to construct their dynamic auditing scheme based on zero-knowledge proof, which is similar to our index switcher in terms of index differentiation and avoidance of tag re-computation. But their design mainly focuses on data dynamics support, while our scheme goes further by achieving dynamic operations support and fair arbitration together. Recently, providing fairness and arbitration in auditing schemes has become an important trend, which extends and improves the threat model in early schemes to achieve a higher level of security insurance. Zheng et al. [11] construct a fair and dynamic auditing scheme to prevent a dishonest client accusing an honest CSP. But their scheme only realizes private auditing, and is difficult to be extended to support public auditing. Kupcu [12] proposes a framework on top of Erway's DPDP scheme [9], where the author designs arbitration protocols on the basis of fair signature exchange protocols in [13]. Moreover, the author goes further by designing arbitration protocols with automated payments through the use of electronic cash. Compared to these schemes, our work is the first to combine public verifiability, data dynamics support and dispute arbitration simultaneously. Other extensions to both PDPs and PoRs are given in. Chen et al. introduce a mechanism for data integrity auditing under the multi server scenario, where data are encoded with network code. Curtmola et al. propose to ensure data possession of multiple replicas across the distributed storage scenario. They also integrate forward error-correcting codes into PDP to provide robust data possession in . Wang et al. utilize the idea of proxy re-signatures to provide efficient user revocations, where the shared data are signed by a group of users. And in they exploit ring signatures to protect the identity-privacy of signers from being known by public verifiers during the auditing

3 System Model

As system model involves four different entities: the data owner/cloud user, who has a large amount of data to be stored in the

cloud, and will dynamically update his data (e.g., insert, delete or modify a data block) in the future; the cloud service provider (CSP), who has massive storage space and computing power that users do not possess, stores and manages user's data and related metadata (i.e., the tag set and the index switcher); the third party auditor (TPAU) is similar to the role of TPA in existing schemes, who is a public verifier with expertise and capabilities for auditing, and is trusted and payed by the data owner (but not necessarily trusted by the cloud) to

assess the integrity of the owner's remotely stored data; the third party arbitrator (TPAR), who is a professional institute for conflict arbitration and trusted by both the owner and the CSP, which is different to the role of TPAU. Cloud users rely on the CSP for data storage and maintenance,

and they may access and update their data. To alleviate their burden, cloud users can delegate auditing tasks to the TPAU, who periodically performs the auditing and honestly reports the result to users. Additionally, cloud users may perform auditing tasks themselves if necessary. For potential disputes between the auditor and the CSP, the TPAR can fairly settle the disputes on proof verification or data update

. Note in following sections, we may use the terms "TPAU" and "auditor" interchangeably, so are the terms "TPAR" and "arbitrator".

- **Cloud**
- **Service**
- **Provider**
- **Owner**
- **users**

2.2 Threat Model

Threat models in existing public auditing schemes [4], [5], mainly focus on the delegation of auditing tasks to a third party auditor (TPA) so that the overhead on clients can be offloaded as much as possible. However, such models have not seriously considered the fairness problem as they usually assume an honest owner against an untrusted CSP. Since the TPA acts on behalf of the owner, then to what extent could the CSP trust the auditing result? What if the owner and TPA collude together against an honest CSP for a financial compensation? In this sense, such models reduce the practicality and applicability of auditing schemes. In a cloud scenario, both owners and CSP have the motive to cheat. The CSP makes profit by selling its storage capacity to cloud users, so he has the motive to

reclaim sold storage by deleting rarely or never accessed data, and even hides data loss accidents to maintain a reputation. Here, we assume the CSP is semi-trusted, namely, the CSP behaves properly as prescribed contract most of the time, but he may try to pass the integrity check without possessing correct data. On the other hand, the owner also has the motive to falsely accuse an honest CSP, e.g., a malicious owner intentionally claims data corruption despite the fact to the contrary so that he can get a compensation from the CSP.

Therefore, disputes between the two parties are unavoidable to a certain degree. So an arbitrator for dispute settlement is indispensable for a fair auditing scheme. We extend the threat model in existing public schemes by differentiating between the auditor (TPAU) and the arbitrator (TPAR) and putting different trust assumptions on them. Because the TPAU is mainly a delegated party to check client's data integrity, and the potential dispute may occur between the TPAU and the CSP, so the arbitrator should be an unbiased third party who is different to the TPAU. As for the TPAR, we consider it honest-but-curious. It will behave honestly most of the time but it is also curious about the content of the auditing data, thus the privacy protection of the auditing data should be considered. Note that, while privacy protection is beyond the scope of this paper, our scheme can adopt the random mask technique proposed in [14], [15] for privacy preservation of auditing data, or the ring signatures in to protect the identity privacy of signers for data shared among a group of users.

2.3 Design Goals

Our design goals can be summarized as follows:

1) Public verifiability for data storage correctness: to allow anyone who has the public key to verify the correctness of users' remotely stored data;

2) Dynamic operation support: to allow cloud users to perform full block-level operations (modification, insertion and deletion) on their outsourced data while guarantee the same level of data correctness, and the scheme should be as efficient as possible;

3) Fair dispute arbitration: to allow a third party arbitrator to fairly settle any dispute about proof

verification and dynamic update, and find out the cheating party.

4 DISPUTE ARBITRATION

4.1 Overview

As we have pointed out before, in the cloud environment, both clients and CSPs have the motive to cheat. In our scheme, the index switcher is used by the auditor to obtain tag indices for requested blocks at proof verification phase, thus the verification result relies on the correctness of the index switcher. However, the generation and update of index switcher are performed by the data owner only, it will potentially give a dishonest owner the opportunity of falsely accusing an honest CSP. In this sense, we must provide some mechanism to ensure the correctness of the index switcher and further the fairness of possible arbitration, so that no party can frame the other party without being detected. A straightforward way is to let the arbitrator(TPAR) keep a copy of the index switcher. Since the change of the index switcher is caused by dynamic operations, the client can send necessary update information (i.e., operation type, operation position, new tag index) to the TPAR for each update operation. With these information, the arbitrator could re-construct the latest version of the index switcher, whose correctness decides the validity of later arbitration. However, such a solution costs $O(n)$ storage at the arbitrator side and needs the arbitrator to be involved in each update operation. Ideally, we want the TPAR only undertake the role of an arbitrator who involves only at dispute settlement, and maintains a constant storage for state information, i.e., public keys of the client and the CSP. As an alternative, we employ the signature exchange idea in [12] to ensure the correctness of the index switcher. Specifically, we rely on both parties exchanging their signatures on the latest index switcher at each dynamic operation. To resist replay attacks, a sequence number indicating the update times is embedded in the signature. A basic fact is that when the client initially uploads his data to the cloud, the cloud needs to run the Commitment to check the validity of outsourced blocks and their tags, and afterwards their signatures on the initial index switcher are exchanged. If this initial signature exchange fails, the client would not assume his data being successfully uploaded. On the other hand, the initial tag index sequence is the same as the block index sequence, that is, the index switcher can be denoted as $\{(i; i)\}1 \leq i \leq n$. Hence, this step of signature

exchange, according to our design, can be easily completed since the initial content of

the index switcher is public to both parties, which is a basis for later signature exchanges. In this sense, our arbitration does not need the existence of a fair signature exchange protocol in [12]. Moreover, because the change of the index switcher is caused by data update operations, the CSP can re-construct the latest index switcher as long as necessary update information (i.e., *op; k; t′ k* in each update record) are sent to the CSP upon each update, which enables the CSP to check the client's signature and generate his own signature on the updated index switcher. Now, upon each data dynamic operation, besides verifying `the updated blocks and tags, the CSP also checks client's signature on the updated index switcher. If succeeds, the CSP sends his signature on the updated index switcher to the client for storage. Then for each successful update, each party holds the other party's signature on the updated index switcher. Such a signature exchange implies an agreement has been reached on the new metadata by the two parties, which is necessary for later dispute resolution. Moreover, the signature is generated on the concatenation of a sequence number *seq* and the index switcher Ω, where *seq* is a monotonically increasing integer that is incremented by one each time. Generally, a dispute may be caused by the disagreement on the proof (including an updated block *m′ k* and its tag *_′ k* in an update request), or on the exchanged signature on the index switcher. According to the time a dispute occurs, we

divide the arbitration occasion into three cases.

- **Case 1:** The dispute occurs when an auditor claims a failure of proof verification during an auditing.
- **Case 2:** The dispute occurs when the CSP receives an invalid update request *up req* from the client.
- **Case 3:** The dispute occurs when the client receives an invalid response to *up req* from the CSP.

Case 1 only involves the disagreement of proof verification, it occurs after a previous successful update where an agreement on the index switcher has been made. While case 2 and case 3 occur before the completion of the current round of update and signature exchange, so the TPAR should be engaged in the protocol to arbitrate on the dispute and help to finish the update and signature exchange.

4.2 Arbitration on Integrity Proof

Let *Sigc* = *Sigskc* (*seq;* Ω) and *Sigs* = *Sigsks* (*seq;* Ω) denote client and CSP's signatures on the index switcher in the last successful update, where *seq* refers to the latest sequence number. When a successful signature exchange completes, the client has the signature *Sigs* of the server, and the server has the signature *Sigc* of the client. During the arbitration, *seqc* and *seqs* denote the sequence number sent by the client and the CSP, Ω*c* and Ω*s* denote the index switcher sent by them, respectively. We assume the public key of each party is in some trusted PKI, hence it can be easily obtained by the other party (including the TPAR). And throughout our protocols, we assume the messages transmitted among three parties are in an authenticated secure channel. We first describe the arbitration protocol of case 1, where the dispute only involves proof disagreement. When the client finds a failure of proof verification during an auditing, he contacts the TPAR to launch an arbitration. Since verifying proof validity needs to access the index switcher to get tag indices of challenged blocks, and verifying signatures also needs the index switcher, it is necessary for each party to send the TPAR the latest index switcher he has kept, along with the signature (on the index switcher) signed by the other party. The arbitration protocol proceeds as follows.

1) The TPAR requests *{seqc;*Ω*c; Sigs}* from the client. Then he checks the signature *Sigs* of the CSP. If it is invalid, the TPAR may punish the client for misbehaving; otherwise the TPAR proceeds.

2) The TPAR requests *{seqs;*Ω*s; Sigc}* from the CSP. Then he checks the signature *Sigc* of the client. If the signature does not verify correctly, the TPAR may punish the CSP for misbehaving; otherwise the TPAR proceeds.

3) If *seqc* = *seqs*, then the TPAR requests from the client the challenged set *Q* that causes dispute on proof verification and retransmit it to the CSP to run the auditing scheme. The CSP computes the proof according to **ProofGen** and returns it to the TPAR for verification. The TPAR checks the proof according to **ProofVerify** using the verified index switcher.

4) If there is a mismatch in *seqc* and *seqs*. The TPAR can be sure that the party who gives a smaller sequence number is performing an replay attack, he may punish the cheating party. Specifically, if *seqc* > *seqs*, the client is cheating by replaying an old signature from the CSP; if *seqs* > *seqc*, the CSP is cheating by replaying an old signature from the client. The security of

this protocol relies on the security of the signature scheme used to sign the index switcher, that is, each party has only negligible probability to forge a signature signed with the other party's private key. Therefore, what should be prevented in the protocol is possible replay attacks launched by a malicious party. As we have included a sequence number in the exchaned signature for each update, we can check whether a replay attack is launched or not by sequence number match. If both signatures verify correctly and their sequence numbers match (*seqc* = *seqs*) then we have Ω*c* = Ω*s*. Due to the initial signature exchange on (0;Ω0) in **TagGen** and **Commitment**, there is at least one round of successful signature exchange before a conflict on proof verification occurs.

4.3 Arbitration on Dynamic Update

Case 2 and case 3 involves the failure of a signature exchange in the current round of update, so it is necessary for the TPAR to help to complete the update and signature exchange. To accomplish this, the successfully exchanged signatures in the previous round should be verified to proceed the current round. The first two steps of the protocol is the same as that of the arbitration protocol on integrity proof, the TPAR requests *{seqc;Ωc; Sigs}* from the client and *{seqs;Ωs; Sigc}* from the CSP. If the TPAR finds any invalid signature, he punishes the corresponding party. According to the result of sequence number comparison (*seqc* and *seqs*), we divide the protocol into two situations.

5 CONCLUSION

The aim of this paper is to provide an integrity auditing scheme with public verifiability, efficient data dynamics and fair disputes arbitration. To eliminate the limitation of index usage in tag computation and efficiently support data dynamics, we differentiate between block indices and tag indices, and devise an index switcher to keep block-tag index mapping to avoid tag re-computation caused by block update operations, which incurs limited additional overhead, as shown in our performance evaluation. Meanwhile, since both clients and the CSP potentially may misbehave during auditing and data update, we extend the existing threat model in current research to provide fair arbitration for solving disputes between clients and the CSP, which is of vital significance for the deployment and promotion of auditing schemes in the cloud environment.We achieve this by designing arbitration protocols based on the

idea of exchanging metadata signatures upon each update operation. Our experiments demonstrate the efficiency of our proposed scheme, whose overhead for dynamic update and dispute arbitration are reasonable.

.

REFERENCES

[1] Y. Deswarte, J.-J. Quisquater, and A. Sa¨ıdane, "Remote integrity checking," in *Proc. 5th Working Conf. Integrity and Intl Control in Information Systems*, 2004, pp. 1–11.

[2] D. L. Gazzoni Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer." *IACR Cryptology ePrint Archive, Report 2006/150*, 2006.

[3] A. Juels and B. S. Kaliski Jr, "Pors: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Computer and Comm. Security (CCS07)*, 2007, pp. 584–597.

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Computer and Comm. Security (CCS07)*, 2007, pp. 598–609.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. 14th Intl Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT 08)*, 2008, pp. 90–107.

[6] 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. 14th European Conf. Research in Computer Security (ESORICS 08)*, 2009, pp. 355–370.

[7] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents." *IACR Cryptology ePrint Archive, Report 2008/186*, 2008.

[8] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," *Network, IEEE*, vol. 24, no. 4, pp. 19–24, 2010.

[9] C. Erway, A. K¨upc¸ ¨ u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Computer and Comm. Security (CCS 09)*, 2009, pp. 213–222.

[10] Y. Zhu, H.Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *Proc. ACM Symp.*

Applied Computing (SAC 11), 2011, pp. 1550–1557.

[11] Q. Zheng and S. Xu, "Fair and dynamic proofs of retrievability," in *Proc. 1st ACM Conf. Data and Application Security and Privacy (CODASPY 11)*, 2011, pp. 237–248.

[12] A. K¨upc¸ ¨ u, "Official arbitration with secure cloud storage application," *The Computer Journal*, pp. 138–169, 2013.

[13] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *Proc. 17th Intl Conf. Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT98)*, 1998, pp. 591–606.

[14] C.Wang, Q.Wang, K. Ren, andW. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[15] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreserving public auditing for secure cloud storage," *IEEE Trans. Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[16] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[17] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. 22nd Intl Conf. Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT03)*, 2003, pp. 416–432.

[18] P. A. Bernstein and N. Goodman, "An algorithm for concurrency control and recovery in replicated distributed databases," *ACM Trans. Database Systems*, vol. 9, no. 4, pp. 596–615, 1984.

[19] J. Hendricks, G. R. Ganger, and M. K. Reiter, "Low-overhead byzantine fault-tolerant storage," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, 2007, pp. 73–86.

[20] J. Gray, P. Helland, P. O'Neil, and D. Shasha, "The dangers of replication and a solution," in *ACM SIGMOD Record*, vol. 25, no. 2, 1996, pp. 173–182.

AUTHOR'S DETAILS:

1.S.FARHANA

shaikfarhana796@gmail.com

2. U.SANDHYA

ummadisettysandhya@gmail.com

ASSISTANT PROFESSOR