JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

ADVANCED ALGORITHM FOR PRIVACY PRESERVASING IN REGENERATING CODE BASED CLOUD STORAGE

Research Guide - Dr.Neeraj Sharma[1]

Department of COMPUTER SCIENCE & ENGINEERING, School of Engineering , Sri Satya Sai University of Technology & Medical Sciences, Sehore, MP, India

Research Co-Guide - Dr. B.Kavitha Rani [2]

Department of COMPUTER SCIENCE & ENGINEERING, CMR Technical Campus, Hyderabad, Hyderabad, Telangana 501401,India

Research Scholar - Banoth Anantharam[3]

Department of COMPUTER SCIENCE & ENGINEERING, School of Engineering, Sri Satya Sai University of Technology & Medical Sciences, Sehore, MP, India

Abstract

To protect the data that is outsourced and stored in the cloud, add fault tolerance, data integrity testing, and failure repair to cloud storage. An approach to public auditing and rewriting codes, both of which are gaining popularity, are looked at in detail here. Data stored in the cloud must be correct. — If you want to store outsourced data in the cloud without having to worry about it getting corrupted, you must enable integrity protection, fault tolerance, and fast data recovery. Because they spread data stripes across multiple servers, regenerating codes use less repair traffic than traditional erasure codes. This makes sure that the system can work even if something goes wrong. So, it is looked into how hard it is to remotely check the validity of regenerating-coded data in a real cloud storage environment where corruptions are likely to happen. For a certain regeneration code, we propose and implement a realistic data integrity protection (DIP) system that keeps the code's inherent properties of fault tolerance and repair traffic savings. Using an adversarial Byzantine DIP method, clients can easily check the integrity of randomly chosen pieces of outsourced data to see if they have been changed in a way that is either accidental or on purpose. Random samples of outsourced data can be checked by the customer to make sure they are correct. The way it works is based on a thin-cloud storage model, which lets a wide range of parameters be changed for the best performance and data security. We used a variety of parameter settings in a real cloud storage testbed to figure out the overhead of our DIP method. It is shown that it is both possible and effective to use remote integrity checking in regenerated codes**.**

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

Introduction

Flexible, on-demand data outsourcing service with benefits like making storage management easier, giving everyone access to information regardless of where they are, and not having to buy expensive hardware or software. Because of these benefits, more and more people are using cloud storage. Using public audits that protect their clients' privacy, data hosting services reveal new security risks to their clients' data, which makes people and businesses still feel hesitant. If the data owner loses control over what happens to their outsourced data, it could affect the data's availability and integrity. One way to keep cloud-stored data from getting corrupted is to add fault tolerance, data integrity testing, and failure repair to cloud storage. This is because the repair bandwidth has been cut down while fault tolerance has been kept. Generic code generation is becoming a more common thing to do. Cloud storage must have fault tolerance, data integrity verification, and failure repair so that outsourced data is safe. Regenerating codes are becoming more popular because they are easier to fix and use less bandwidth to do so. If the current ways of regenerating coded data are used, the people who own the data must be online all the time and make any changes themselves. This may cost money in some situations. By changing the classic Merkle Hash Tree construction for block tag authentication, this method improves on existing proof-of-storage models and makes data flow more efficient. Multiple auditing tasks also look into the technique of bilinear aggregate signature in order to use TPA, which can do many auditing tasks at the same time, to expand the result into a multi-user scenario. Both how well they work and how safe they are have been thoroughly tested, and the results are convincing.

Related work

We look at the problem of making sure that static data is correct, which is a common concern in long-term storage systems. A single server scenario is used to evaluate this problem for the first time.

K. S. B. et al. [1] Using proxy and partial keys, we can change data in a cloud storage system that is based on codes that can be changed and a public auditing system that protects privacy. Because of this change, data owners no longer have to give permission to change data. An independent third-party auditor can increase security and give the user access to information about the cloud data being saved. This is a very useful piece of information. Here, we talk in more detail about a public auditing method and regenerating codes, both of which have made it possible to store data securely in the cloud.

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

P. Tan et al[2] The problem of fault tolerance in the cloud storage system has been said to be solved by stream regeneration with regenerating codes. Stream regeneration uses both pipelined regeneration and regeneration codes to get the results that are wanted. Pipelined regeneration takes longer when stream regeneration is used. With the same number of nodes, both stream regeneration and pipelined regeneration use the same amount of network traffic. However, stream regeneration is more efficient at regenerating than pipelined regeneration. Stream regeneration can fix up to r failed nodes in each regeneration round while keeping the same level of data integrity as pipelined regeneration. This is possible because the pipeline doesn't get in the way when the stream is being cleaned up.

J. Zhang, et al[3] This piece will show that their method is not as safe as they say it is. Any data block's authenticator can be faked by the proxy of the data owner, which makes the security of their protocol useless. We'll prove it. After finding the flaw in the design, we have high hopes that this kind of problem won't come up again when new protocols are made.

J. Liu et al[4] The conventional idea for a public auditing system should include a proxy that can make new authenticators. Using partial keys and two or more keys, we can make a unique public authenticator that can be checked and can be made again. Anyone can prove who they are by using this authenticator. So, our method can save data owners the trouble of having to keep up with their online presence. To keep data private, a pseudorandom function is also used to mix up the encode coefficients. Under the idea of a random oracle, our method has been shown to be safe through a lot of security analysis. The evaluation of our method through experiments shows that it is very effective and can be used in a cloud storage system that uses code that has been regenerated.

Pengxu Tan et al[5] This paper showed how to make distributed networked storage work even if something goes wrong. Using threshold public-key encryption, or SRCS, it is possible to protect code that has been regenerated with minimal redundancy and high efficiency. Before storing their data in an SRCS-based storage solution, the people who own the data must first send their private keys to a group of key servers. Even if the attacker gets into all of the storage servers, he won't be able to get to the data. In the semi-adaptive model, SRCS can be used by making the decisional BDHE assumption..

H. C. H. Chen, et al[6] As part of our FMSR codes, we reduce the amount of encoding that storage nodes need to do while they are being fixed. Another important part of the design is network coding in repair, which we also keep. We build a prototype that can be installed in

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

different cloud environments based on the NCCloud proof-of-concept. Our results show that using FMSR codes instead of RAID-6 codes for cloud storage operations like uploading and downloading data saves a lot of money on repair costs.

Kun Huang, et al[7] On top of that, a real-world DPDP scheme was made for the regeneration code and put into place. Because of this, the default protections for data integrity, efficient dynamic updating, fault tolerance, and repair traffic savings have been kept. The innovative Memory Adversary model in our DPDP system was built on the foundation of dynamic operations. There are many settings that can be tweaked to find a good balance between speed and safety. The main goal of this study is to put our technique for cloud storage into action and figure out how much extra work it takes. We show that DPDP is a good choice for use in our particular algorithm for regeneration.

J. Chen, et al[8] The Remote Data Checking and Repairing (RDCR) method used in this study regenerates codes that need the least amount of bandwidth. Our method makes it easier on data owners by letting a third party check that the data they have given is correct. Our method, unlike those that came before it, allows for the accurate correction of wrong data, which means that less computing is needed overall. We put our plan into action, and the tests show that RDCR costs less to process and communicate than other systems that are currently being used..

Proposed methodology

The paradigm described in this article is made up of three parts: the cloud server, the group of users, and the public verifier. A group can have two kinds of users: the first person who joined the group and a group of other people. The first person to use shared data shares it with other people in the same group. All members of the group, as well as the original user, are included. All group members have access to all shared information and can read, change, or delete it. On the cloud server, both the shared content and the information used to check it are kept (also known as signatures). Public verifiers of the integrity of shared data are third-party auditors who specialise in providing professional auditing services or a data user from an outside organisation who plans to use shared data stored on a cloud server. This kind of public verification can be done by a user of the data. First, a public verifier will send an auditing challenge to the cloud server to make sure that the data being sent is correct. After having to deal with the challenge of auditing,

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

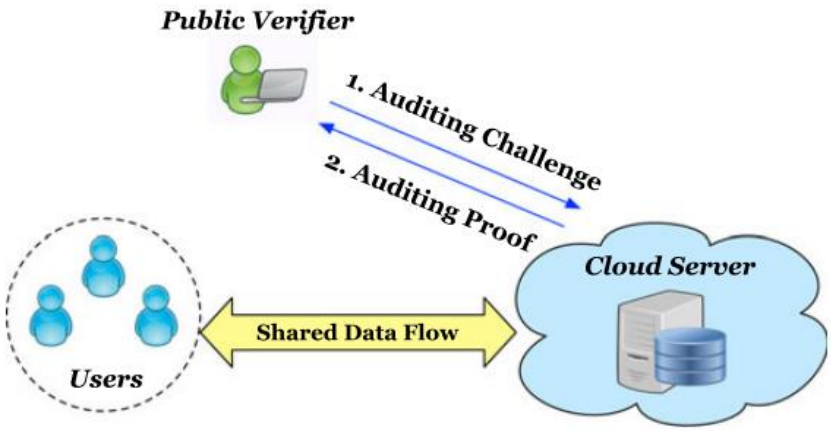

Fig. 1. As part of our system concept, we've paired the cloud server with a group of users.

The cloud server tells the public verifier that it has the shared data by sending an auditing proof. The public verifier then checks to see if the auditing proof is correct before deciding if the whole set of data can be trusted. A public audit of the system is done using an authentication challenge and response protocol between a public verifier and the cloud server. [9]

Threat Model Integrity Threats. There are two possible threats that need to be stopped in order to keep the data that is sent safe. First, an opponent might try to make sure that the information given isn't true. Data that is stored in the cloud could be damaged or even lost if something goes wrong with the technology or with a person. Worse, the cloud service provider may not want to tell customers about corrupted data because it could hurt its reputation and cause it to lose money from the services it provides. Because of this, things look a lot more dangerous. Privacy is not always safe. Because this information is private and confidential, the group will not know who signed each block of shared material. Auditing can show who signed each block of data shared by a public verifier, whose job is only to check the integrity of shared data. To make sure that shared data is correct, a public verifier can only confirm that it is correct. After the public verifier publishes the identity of the signer on each block, it is easy to tell a high-value target from other targets. This could be a unique user in a group or a unique block in shared data.

There are different ways that could be taken. As an alternative, you could ask everyone in the group to share a global private key. This way, during the public auditing process, the identity of the person who signed each block can be kept. After that, each user will be able to sign

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

blocks with the global private key. When a group member is hacked or leaves the group, a new global private key must be made and shared with the rest of the group in a safe way. This makes it very hard for users to keep track of their keys and share them. With our method, users can keep using their own private keys to compute verification metadata without having to make or share new secret keys. Because our solution doesn't need any extra secret keys. Adding a trustworthy proxy to the system model is another way to protect user identities while using cloud storage. To put it another way, this trusted third party is the one who gets all of the members' data, signs it, and sends it to the cloud. This means that after this, a public verifier can only confirm that the proxy signed the data. They can't find out who else is in the group. This solution is less safe than others because there is only one place where something could go wrong: the proxy. Also, it's possible that not all of the members of the group are comfortable letting the same proxy sign documents and upload data on their behalf. Using a group signature, which is another option, is another way to hide your identity. Our recent study shows that the question of how to think about a public auditing system based on group signatures is still not solved. The design goals of our technology can be met in this way as well as through Trusted Computing. Direct anonymous attestation, which the Trusted Computing Group has chosen as the anonymous method for remote authentication in the trusted platform module, lets users share data with a public verifier without revealing their identity. In other words, users have the choice to do so. One big problem with this method is that each person who uses it needs their own equipment that was made just for them. Even worse, the cloud service provider would have to move all of its current cloud services to a secure computing environment, which would be expensive and impossible.

Privacy models Privacy models

We can deal with the trade-off between privacy and utility by putting privacy first and making an ex ante privacy promise that the anonymized data set must meet. This is true no matter how the data was hidden or what it was in the first place. Most of the time, a certain privacy model can be met with a number of different masking techniques. You need to use the right kind of masking to get the most out of your data (because satisfying the model already ensures privacy). Users, such as the people in charge of the local proxy's security, can get a better idea of how and how well the data that were sent to the cloud were protected, no matter how many, what kind, or how they were organised.

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

k-anonymity is the oldest and most well-known model for syntactic privacy. It tries to hide each subject in a bigger group of k other subjects so that records can't be used to find them again. In order to reach this goal, the k-anonymity criterion says that every k-minus-1 records in the anonymized data set must be the same when it comes to the quasi-identifier attributes. Since no subject's identity can be linked to less than k records in a k-anonymous dataset, the chance of a correct re-identification is no more than 1/k in such a collection. When k is greater than , you can be sure that there won't be any re-identifications if the chance of re-identification is less than . Changes to the value of k can also be used to change how much exposure each person gets. As part of the model's k-anonymity requirement, the quasi-identifier values of records must be changed so that they can't be told apart. To make sure that k-anonymity is kept, the following masking methods are often used:

• In order to increase the number of entries that share a certain combination of quasi-identifier attribute values, the most unusual attribute values are replaced with values that are missing. All of the cluster's entries are replaced with the same generalisation, which is a quasi-identifier attribute value that all of the cluster's records share.

• Through microaggregation, k or more clusters of records are made, and each record's quasi-identifier attributes are replaced with the centroid or average value of its cluster. A quasi-identifier attribute value is used to categorise records in accordance with similarities in quasi-identifier attribute values. When you use common values instead of quasi-identifiers, you lose less information (and usefulness) in the process. Even though k-anonymity protects against identity disclosure, confidential attribute values (like a diagnosis) can still be revealed if there isn't much variation in those values (like how likely it is to be able to re-identify the person the record is about). As an example, if the difference between the secret attribute values in a group of k records is small, then the subject's diagnosis may be known (e.g., all individuals in the cluster have cancer). The l-diversity and t-closeness parameters of the k-anonymity model can be changed to make it less likely that an attribute will be revealed. To meet l-diversity requirements, at least k records that are the same must have secret attribute values that are different enough from each other to prevent the clear disclosure of sensitive information. T-Closeness, which says that the way secret attribute values are spread out in each cluster should be similar to how they are spread out in the whole data set, backs up this idea even more. This model gives the best guarantee of privacy of all the ones that are related to k-anonymity. To meet the t-Closeness criteria, generalisation and microaggregation were used. Based on t-closeness and quasi-identifier attributes, records are grouped so that: the

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

distribution of secret values within the cluster is different enough to meet this need; and the similarity between records in a cluster that meets this need can be maximised.

Results analysis

We timed how long it took to do three different tasks: file encoding, file verification, and file regeneration. This helped us figure out how well our planned local cloud storage system worked. We also look at how much longer this method takes to run compared to other options. Experiments in MATLAB R2014b were done on a laptop computer with an Intel Core i3 CPU and 3 gigabytes of RAM. Figure 1 shows an example of how long it takes to encode files. The process of making encoded matrices is easy and straightforward because it doesn't require any math. In the 50 MB document file we used, there were a number of sub-sections. On average, it takes three minutes and sixty-four seconds to encode a file.

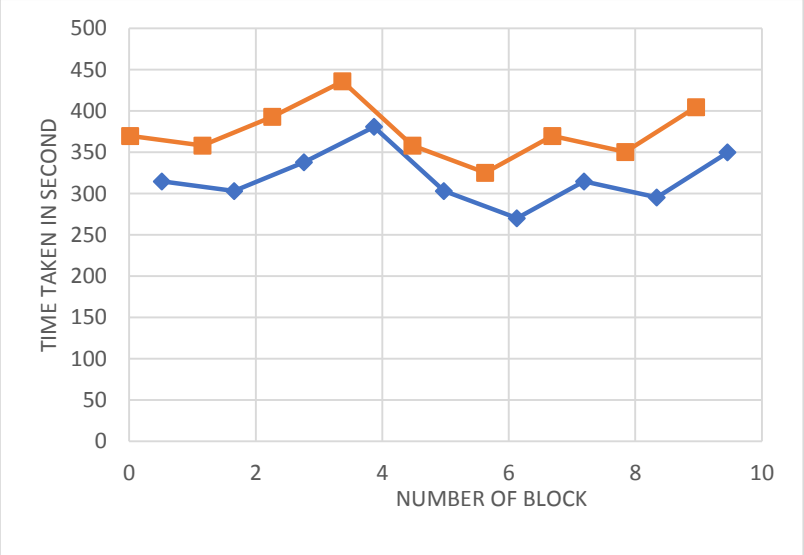

Fig. 1. Number of Blocks in Challenge

, Here, you can see how long the Verification operation has been going on for. The blocks that are needed to finish the assignment are switched around. We think that this process takes an average of 2.55 seconds. Figure 1 shows how long it takes for the process of regeneration to finish. It doesn't need to be updated because this fix is correct and makes the programme easier to work with. For this experiment, we used a 50 MB doc file with a variable number of blocks. The average time it takes to do a regeneration procedure is 0.189 seconds. we compare the total amount of time it takes our method and other methods to run.

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

Conclusion

Using a cryptographic hash, we show a new way to make sure that data stored in the cloud is correct and available. We came up with this way. After we were done with our security investigation, it was easy to show that our security measures work against different types of attacks. Our proposed method was also tested and compared to other methods. The results showed that our method took much less time to use than the other methods. This technique makes the cloud storage service more secure by guaranteeing data integrity and availability while reducing the amount of time wasted on tasks that aren't necessary.

Reference

[1]., K. S. B. "Regenerating-Code-Based Cloud Storage Using Privacy-Preserving Public Auditing". *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 3, no. 12, Dec. 2015, pp. 6697-8, doi:10.17762/ijritcc.v3i12.5122.

[2].P. Tan, Y. Chen, C. Li and G. Yang, "Stream Regeneration with Regenerating Codes for the Fault-Tolerant of Cloud Storage," 2012 Second International Conference on Business Computing and Global Informatization, 2012, pp. 735-738, doi: 10.1109/BCGIN.2012.197.

[3].J. Zhang, R. Lu, B. Wang and X. A. Wang, "Comments on "Privacy-Preserving Public Auditing Protocol for Regenerating-Code-Based Cloud Storage"," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 1288-1289, 2021, doi: 10.1109/TIFS.2020.3032283.

[4].J. Liu, K. Huang, H. Rong, H. Wang and M. Xian, "Privacy-Preserving Public Auditing for Regenerating-Code-Based Cloud Storage," in IEEE Transactions on Information Forensics and Security, vol. 10, no. 7, pp. 1513-1528, July 2015, doi: 10.1109/TIFS.2015.2416688.

[5].Pengxu Tan, Yue Chen and Chaoling Li, "A secure regenerating code for the fault-tolerant of distributed networked storage," 2013 IEEE 4th International Conference on Software Engineering and Service Science, 2013, pp. 507-510, doi: 10.1109/ICSESS.2013.6615360.

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

[6]. H. C. H. Chen, Y. Hu, P. P. C. Lee and Y. Tang, "NCCloud: A Network-Coding-Based Storage System in a Cloud-of-Clouds," in IEEE Transactions on Computers, vol. 63, no. 1, pp. 31-44, Jan. 2014, doi: 10.1109/TC.2013.167.

[7]. Kun Huang, Jian Liu, Ming Xian, H. Wang and Shaojing Fu, "Enabling dynamic proof of retrievability in regenerating-coding-based cloud storage," 2014 IEEE International Conference on Communications Workshops (ICC), 2014, pp. 712-717, doi: 10.1109/ICCW.2014.6881283.

[8]. J. Chen, Y. Peng, R. Du, Q. Yuan and M. Zheng, "Regenerating-Codes-Based Efficient Remote Data Checking and Repairing in Cloud Storage," 2015 IEEE Trustcom/BigDataSE/ISPA, 2015, pp. 143-150, doi: 10.1109/Trustcom.2015.368.

[9]. J. Li, T. Li and J. Ren, "Beyond the MDS Bound in Distributed Cloud Storage," in IEEE Transactions on Information Theory, vol. 66, no. 7, pp. 3957-3975, July 2020, doi: 10.1109/TIT.2020.2993038.

[10]. N. Arya, R. R. Rout and G. Lingam, "Network Coding Based Multiple Fault Tolerance Scheme in P2P Cloud Storage System," 2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS), 2018, pp. 223-228, doi: 10.1109/ICIINFS.2018.8721316.

[11]. J. Li, T. Li and J. Ren, "Beyond the MDS bound in distributed cloud storage," IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 307-315, doi: 10.1109/INFOCOM.2014.6847952.

[12]. D. Hadke and R. Babu, "Privacy-Preserving and Public Auditing for Regenerating-Code-Based Cloud Storage Using Finger Print Authentication," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020, pp. 1284-1288, doi: 10.1109/I-SMAC49090.2020.9243362.

[13]. H. C. H. Chen and P. P. C. Lee, "Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage: Theory and Implementation," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, pp. 407-416, Feb. 2014, doi: 10.1109/TPDS.2013.164.

[14]. Y. Hu, P. P. C. Lee and K. W. Shum, "Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems," 2013 Proceedings IEEE INFOCOM, 2013, pp. 2355-2363, doi: 10.1109/INFCOM.2013.6567040.

[15]. Y. Hu, P. P. C. Lee and K. W. Shum, "Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems," 2013

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

Proceedings IEEE INFOCOM, 2013, pp. 2355-2363, doi: 10.1109/INFCOM.2013.6567040.

[16]. J. Wang, Z. Yan, K. -C. Li, H. Xie and X. Liu, "Local Codes With Cooperative Repair in Distributed Storage of Cyber-Physical-Social Systems," in IEEE Access, vol. 8, pp. 38622-38632, 2020, doi: 10.1109/ACCESS.2020.2975577.

[17]. Kai He, Chuanhe Huang, Jiaoli Shi and Jinhai Wang, "Public integrity auditing for dynamic regenerating code based cloud storage," 2016 IEEE Symposium on Computers and Communication (ISCC), 2016, pp. 581-588, doi: 10.1109/ISCC.2016.7543800.

[18]. J. Li, T. Li and J. Ren, "Secure regenerating code," 2014 IEEE Global Communications Conference, 2014, pp. 770-774, doi: 10.1109/GLOCOM.2014.7036901.

[19]. G. Calis, S. Shivaramaiah, O. O. Koyluoglu and L. Lazos, "Repair Strategies for Mobile Storage Systems," in IEEE Transactions on Cloud Computing, vol. 9, no. 4, pp. 1575-1591, 1 Oct.-Dec. 2021, doi: 10.1109/TCC.2019.2914436.

[20]. B. Zhu, K. W. Shum, H. Li and H. Hou, "General Fractional Repetition Codes for Distributed Storage Systems," in IEEE Communications Letters, vol. 18, no. 4, pp. 660-663, April 2014, doi: 10.1109/LCOMM.2014.030114.132694.

[21]. N. -E. Alouane, J. Abouchabaka and N. Rafalia, "A thick-cloud solution for data auditing in a cloud environment," 2016 International Conference on Electrical and Information Technologies (ICEIT), 2016, pp. 8-15, doi: 10.1109/EITech.2016.7519562.

[22]. C. M. Geeta et al., "SRCBT: Secure Regeneration of Corrupted Blocks by TPA in Cloud," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 835-838, doi: 10.1109/TENSYMP50017.2020.9230878.

[23]. T. Zhou, H. Li, B. Zhu, Y. Zhang, H. Hou and J. Chen, "STORE: Data recovery with approximate minimum network bandwidth and disk I/O in distributed storage systems," 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 33-38, doi: 10.1109/BigData.2014.7004381.

[24]. Kumar, R., Singh, J.P., Srivastava, G. (2014). Altered Fingerprint Identification and Classification Using SP Detection and Fuzzy Classification. In: , et al. Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012. Advances in Intelligent Systems and Computing, vol 236. Springer, New Delhi. https://doi.org/10.1007/978-81-322-1602-5_139

ISSN- 2394-5125 VOL 7, ISSUE 19, 2020

[25]. Gite S.N, Dharmadhikari D.D, Ram Kumar," Educational Decision Making Based On GIS" International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-1, Issue-1, April 2012.

[26]. Ram Kumar, Sarvesh Kumar, Kolte V. S.," A Model for Intrusion Detection Based on Undefined Distance", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1 Issue-5, November 2011

[27]. Vibhor Mahajan, Ashutosh Dwivedi, Sairaj Kulkarni,Md Abdullah Ali, Ram Kumar Solanki," Face Mask Detection Using Machine Learning", International Research Journal of Modernization in Engineering Technology and Science, Volume:04/Issue:05/May-2022

[28]. Kumar, Ram and Sonaje, Vaibhav P and Jadhav, Vandana and Kolpyakwar, Anirudha Anil and Ranjan, Mritunjay K and Solunke, Hiralal and Ghonge, Mangesh and Ghonge, Mangesh, Internet Of Things Security For Industrial Applications Using Computational Intelligence (August 11, 2022). Available at SSRN: https://ssrn.com/abstract=4187998 or http://dx.doi.org/10.2139/ssrn.4187998

[29]. Kumar, Ram and Aher, Pushpalata and Zope, Sharmila and Patil, Nisha and Taskar, Avinash and Kale, Sunil M and Gadekar, Amit R, Intelligent Chat-Bot Using AI for Medical Care (August 11, 2022). Available at SSRN: https://ssrn.com/abstract=4187948 or http://dx.doi.org/10.2139/ssrn.4187948

[30]. Ram Kumar, Manoj Eknath Patil ," Improved the Image Enhancement Using Filtering and Wavelet Transformation Methodologies", Turkish Journal of Computer and Mathematics Education ,Vol.13 No.3(2022), 6168-6174.

[31]. Ram Kumar, Jasvinder Pal Singh, Gaurav Srivastava, "A Survey Paper on Altered Fingerprint Identification & Classification" International Journal of Electronics Communication and Computer Engineering ,Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209.