ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

Comprehensive Review of Local and Cloud File Synchronization Algorithms with Efficient Conflict Resolution

Nikita Ramesh Ubarhande[1], Prof. S. M. Dandge[2]

[1]Student, CSE, Pankaj Laddhad Institute of Technology and Management, Buldhana, India

[2]Assistant Professor & HOD, CSE, Pankaj Laddhad Institute of Technology and Management, Buldhana, India

Abstract: *Making sure that two or more sites have the same current files is known as file synchronization. For effective file management between local and cloud storage systems, it is crucial. However, a number of issues, including network latency, capacity restrictions, file conflicts, and user preferences, can make file synchronization difficult. In this research, we offer a complete file synchronization algorithm for easy local and cloud file management with conflict resolution that can handle many circumstances and requirements. A file system monitor, a file transfer manager, a file conflict detector, and a file conflict resolver make up our algorithm. We employ a number of criteria, including synchronization time, data transfer, and user satisfaction, to assess the efficiency and performance of our method. We show the benefits and drawbacks of our approach by contrasting it with other file synchronization options. We also talked about the obstacles and future directions of file synchronization research.*

Keywords: File Synchronization, Cloud storage, Conflicts, Algorithm, Data Transfer.

I. INTRODUCTION

One typical use of networking technology is the sharing of digital files over a network. You can share files with: a) Users and machines (e.g., when a file is downloaded from or uploaded to a server); b) Machines (e.g., automated backups); and c) Different users (either directly through P2P file sharing services or through machines, such as uploading a file to a server from which the other party can download it). These days, consumers frequently use synchronization services like Dropbox [1] or Google Drive [2] to share data across their own devices linked via a network. Typically, they accomplish this by enabling users to upload files from one device to central servers and then receive those files from other devices that belong to the same user. Keep in mind that users can also choose to make their files public or share them with others.

The files are divided into chunks (or pieces) using peer-to-peer (P2P) synchronization systems, which subsequently replicate the data on a subset of peers. Users' files are stored on a central server that is owned and controlled by a certain entity (such as a small business or an enterprise) using a cloud-based synchronization system (also known as a cloud-based storage service). From one device, users post files to this server, which they then download from another (or the same device, if they misplace the original file). Additionally, users can share their data with others. Depending on the service offered, a cloud-based synchronization service can be expanded to give users a platform for collaboration. Web and native application development technologies are used as the front-end of these services, which are offered on numerous platforms.

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

In order to facilitate a smooth interface with the real cloud drive, several of them offer desktop programs that function as drives attached to the computer. These services typically use a freemium business model, where customers can upgrade to a higher plan with more capacity and extra features after receiving a defined amount of initial storage for free with a limited feature set. [8] provides a useful comparison of some of the most well-known cloud storage and synchronization services. Users may access and utilize cloud services much more easily with such a strategy.

A. Objectives

- To ensure that changes made to files are synchronized across all devices in real-time or near real-time, minimizing delays in accessing the latest version of a file.
- To implement mechanisms to detect that arise when the same file is edited on multiple devices simultaneously, ensuring that data integrity is maintained.
- To optimize the synchronization process, minimizing the amount of data transferred and reducing synchronization times.
- To access previous versions of files and open them if needed.
- To scale efficiently as the number of users and the time of conflict.

B. Challenges of file Synchronization

Businesses whose employees use consumer-grade apps to access company files frequently have security worries about file syncing. These programs might not only produce "shadow IT," but they might also have flaws that hackers might use to compromise the file and the company. Loss of centralized control is another issue with file syncing. Business files are frequently synced to personal devices by remote and hybrid teams, particularly if the company has a "bring your own device" policy in place. These teams' files will remain accessible on their devices even if members depart the company. It's possible that the organization won't be able to manage the files on the former employee's device, let alone remove them to safeguard company information. Another issue with file syncing is unintentional movements and deletions. It's crucial to employ file sync software that allows file recovery in order to allay these worries. In order to help managers better manage files and stop unintentional movements and deletions, the software should also offer features like audit logs, notifications, and granular permissions.

II. LITERATURE SURVEY

According to Csirmaz et al., it is a common task to synchronize divergent copies of some data that are kept on many devices and/or at separate locations. Numerous theoretical and practical publications addressing this issue have been produced during the past 20 years. File synchronization is a feature that backup software typically has to make managing and recovering data easier. Cloud services are typically used to provide file synchronization. Dedicated file synchronization solutions sometimes include extra capabilities to facilitate file sharing and collaboration with stored files and documents in addition to managing the preserved data. Because the service front-ends are so effectively integrated into web clients and desktop and mobile environments, the end-user may effortlessly access these

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

cloud storage services. The intricate and complicated service back-ends are concealed by straightforward user interfaces. The "cloud storage" environment often incorporates collaboration features. For instance, Dropbox Paper is an extension of Google Drive storage, Office 365 is integrated with One Drive storage, and Google Docs is an application layer integrated into Google Drive storage [1].

According to Dürsch et al., applications that allow digitally conducting QDA are categorized as computer assisted qualitative data analysis software. QDAcity1 is one example of computer-assisted qualitative data analysis software (CAQDAS). The Professorship for Open Source Software at the Friedrich-Alexander-Universität Erlangen-Nürnberg created and runs QDAcity, a cloud-based web platform that enables several analysts or researchers to work together on QDA. Allowing researchers to exchange and discuss various interpretations, concepts, and conclusions can be highly advantageous for the QDA process because it deals with large amounts of ambiguous and subjective data. Other fields similarly employ the strategy of "shrinking the feedback loop" and fostering close collaboration to improve a process. Examples of this include agile software development methodologies such as Extreme Programming (Beck, 2000). But as of right now, QDAcity simply permits numerous researchers to work together simultaneously on a shared topic; it does not provide more detailed collaboration on a shared document. One famous example of intimate, digitally enabled cooperation is real-time collaborative editing of a shared document. [2]

According to Martins et al., the cloud has grown commonplace in recent years. These solutions are used by several apps and services that have consumers all around the world. Social media and other cloud applications with a large worldwide user base typically use distributed databases that put low latency ahead of high consistency. These techniques eliminate the need for coordination, which would penalize performance in a communication process that can span continents by requiring reads and writes to reach most replicas. These kinds of applications typically operate across several datacenters in addition to database replicas.

In order to provide quick response times for users from all over the world, instances are typically geo-replicated. As the number of copies increases, it also becomes crucial to divide data in a way that maintains the replication's fault tolerance guaranties. This is because having every piece of data in every database copy may not be required and can undoubtedly become very costly. Having database replicas close to the users is insufficient in this situation. Achieving the intended low latency also requires minimizing coordination between replicas executing reads and writes. Let's say you have a local server near a client. A client database action would need to communicate with most database instances in order to finish. The intended low latency would be totally broken by this. Weak consistency models have become more and more prevalent as a result. [3]

III. PROPOSED SYSTEM

A. Proposed System

File Synchronization Algorithm: To find the differences between two files, file comparison methods are utilized. File comparison can be done in a number of ways, each with pros and cons of its own. These are some typical algorithms:

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

Step 1: Byte-by-byte comparison: This is the most basic type of comparison in which every file byte is compared. Although it is quick, it doesn't offer comprehensive details about the variations.

Step 2: Line-by-line comparison: This method works well for text files because it compares files line by line. It may not be able to identify changes inside lines, but it can indicate lines that have been added or removed.

Step 3: Token-based comparison: This technique analyzes the files by breaking them up into tokens (words, phrases, etc.). Tokenization logic is needed, but it is more flexible than line-by-line comparison.

Step 4: Tree-based comparison: A tree-based method can be applied to structured data, such as XML or JSON. It highlights variations at various granularities by comparing the content and structure of the trees.

Step 5: Hybrid approaches: Depending on the type of files being compared, several tools combine various methods to balance accuracy and speed.

Working of Checksums

In order to guaranty file integrity and identify discrepancies, checksums are frequently utilized in file synchronization and dispute resolution. Checksums can be used in the following ways: Creating Checksums: Every file is given a checksum, which is a distinct identifier determined by the content of the file. MD5, SHA-1, and SHA-256 are common checksum algorithms. Comparing Checksums: To ascertain whether two files are identical, checksums are compared during file synchronization. The files are regarded as identical if the checksums match. The files contain different contents if the checksums are different. Checksums can be used to resolve conflicts when there are conflicting changes. For instance, checksums can be used to compare the differences between two users' modifications to the same file. Verification of Checksums: During synchronization, checksums can also be used to confirm the integrity of files. To verify that a file was not corrupted during transmission, the checksum of the file can be recalculated after transfer and compared to the original. Efficiency: Because checksums offer a fixed-size representation of the file's content, they are effective at identifying changes in huge files. Because of this, comparing checksums is simpler than comparing huge files' complete contents.

Checksums have limits even if they are helpful for identifying file differences. For instance, using contemporary techniques, checksum collisions—different files having the same checksum—are feasible but uncommon. Furthermore, checksums simply indicate that a file's contents have changed; they do not reveal the precise changes that have place in the file. Upper layer protocols use checksum as its error detection technique, which is thought to be more dependable than LRC, VRC, and CRC. This technique uses a checksum generator on the sender side and a checksum checker on the recipient side.

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

B. Flowchart

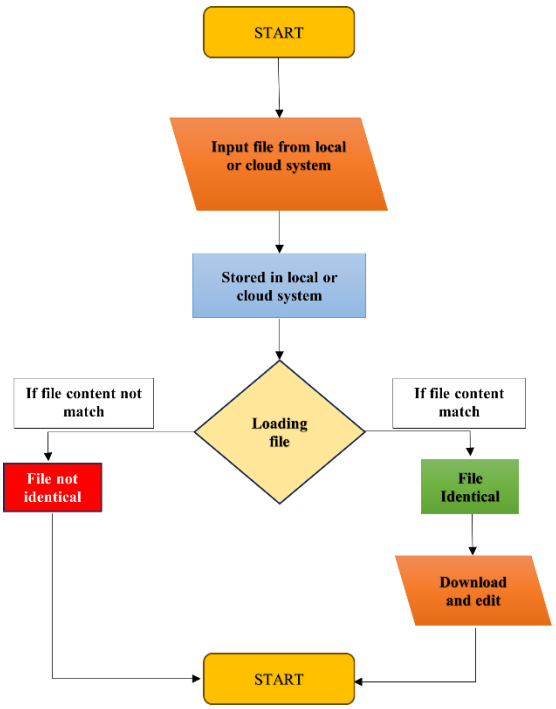


Figure 1: Flowchart of the Key Agreement Protocol Mechanism

C. Libraries Used

OS Module: This module is crucial for us to implement the directory functionality of the file system since we are working with files on the local system in our suggested system. Python's OS module offers functions for communicating with the operating system. Python's standard utility modules include OS. A portable method of utilizing operating system-dependent functionality is offered by this module. Numerous file system interaction functions are included in the os and os.path modules. The Amazon Web Services (AWS) SDK for Python, known as Boto3, enables Python developers to easily communicate with AWS services such as Amazon S3 and Amazon EC2. By providing a Pythonic programming approach to cloud resource management, Boto3 streamlines your interactions with AWS.

The Python time module, as its name implies, enables the manipulation of time in Python. It enables features like obtaining the current time and stopping the program's execution, among other things. Therefore, we must import this module before we can begin.There is no need to install the time module separately because it is included with Python's standard utility module. In order to provide the time when the uploaded file was altered or opened, our system uses timestamp technology. JSON JavaScript Object in Python One way to organize data is through notation. Data storage and transport between the browser and the server are its primary uses.Python has a built-in JSON library that handles JSON as well. All of the tools required to work with JSON objects are included in this package, including parsing, serializing, deserializing, and many more.

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

Hashlib: A function that accepts input data and generates a statistically distinct output that is specific to that collection of data is known as a cryptographic hash function. A fixed-length byte stream called a hash is employed to guaranty the data's integrity. This article will teach you how to use the hashlib module in Python to get a file's hash. There is no need to install the hashlib module because it is a built-in module that is included by default with Python's standard library. A standard interface for numerous secure cryptographic hash and message digest algorithms is implemented by the hashlib module. For every kind of hash, there is a single constructor method. Each yields a hash object with the same straightforward interface. This module always contains hash algorithm constructors.

D. MD5 Hash

Hash functions are widely employed in cryptographic algorithms, digital signatures, fingerprints, password storage, and many more real-world applications. As Python programmers, we require hash functions to store passwords in databases, verify data integrity while sending data over a public network, and check for duplicate data or files. We utilize a hash function in our project that takes a sequence of bytes and outputs a 128-bit hash value. This function is typically used to verify data integrity, but it has security flaws.

IV. CONCLUSION AND FUTURE SCOPE

A. Conclusion

The checksum method outperforms all other file synchronization techniques for files with the highest degree of similarity. Similar file types benefit more from de-duplication; selecting the optimal chunk size may result in the greatest number of chunks. Therefore, the chunk size determines the likelihood of obtaining the greatest number of comparable chunks. Finding the ideal chunk size is therefore a challenge in and of itself, as it changes depending on the kind of files that are synchronized. By detecting changes in the content and alerting the user before opening it, the proposed solution partially addresses the file synchronization problem. Reconstructing the same file using the same chunk on the destination device presents another challenge. The chunking method outperforms all other file synchronization techniques. In addition to various techniques for file synchronization,

B. Future Scope

- AI-Based Synchronization: Future systems can use machine learning and AI to predict file changes, synchronization requirements, and potential conflicts before they occur.
- Intelligent Conflict Resolution: Advanced algorithms can automatically identify the type and importance of conflicts and select the most appropriate resolution strategy.
- Real-Time Synchronization: Future research can focus on faster and more reliable real-time synchronization across multiple local devices and cloud platforms.
- Enhanced Security and Privacy: Integration of end-to-end encryption, secure authentication, access control, and privacy-preserving synchronization can improve data protection.
- Edge and Distributed Computing: Synchronization can be extended to edge devices and distributed environments to reduce latency and dependence on centralized cloud servers.

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

- Cross-Platform Support: Future algorithms can provide seamless synchronization across Windows, Linux, macOS, smartphones, and different cloud-storage providers.
- Bandwidth and Energy Optimization: Adaptive synchronization techniques can minimize bandwidth consumption, storage requirements, and energy usage, particularly for mobile and IoT devices.
- Blockchain-Based File Integrity: Blockchain or distributed ledger technologies could be investigated for maintaining tamper-evident synchronization records and file-version histories.
- Scalable Multi-User Synchronization: Future work can address synchronization challenges involving large numbers of users, devices, shared files, and geographically distributed cloud environments.
- Autonomous File Management: Future systems could combine AI, version control, and synchronization algorithms to automatically organize, prioritize, backup, and resolve file changes with minimal user intervention.

REFERENCES

[1] Elod P. Csirmaz and Laszlo Csirmaz, "Synchronizing Many Filesystems in Near Linear Time", arXiv:2302.09666v2 [cs.IT] 17 May 2023

[2] Martin Dürsch, "Scaling Real-time Collaborative Editing in a Cloud-based Web App", Erlangen, 19 April 2023

[3] João Gonçalves Martins, "Query Processing in Cloud Databases with Partial Replication" NOVA University Lisbon March, 2023

[4] Masoumeh Hajvali, Sahar Adabi, Ali Rezaee and Mehdi Hosseinzadeh, "Decentralized and scalable hybrid scheduling-clustering method for real-time applications in volatile and dynamic Fog-Cloud Environments" (2023) 12:66, https://doi.org/10.1186/s13677-023-00428-4, Journal of Cloud Computing: Advances, Systems and Applications

[5] Elod P. Csirmaz 1,_ and Laszlo Csirmaz, "Data Synchronization: A Complete Theoretical Solution for Filesystems" Future Internet 2022, 14, 344. https://doi.org/10.3390/fi14110344

[6] Novak Boˇskov, Ari Trachtenberg, and David Starobinski. Enabling costbenefit analysis of data sync protocols, 2023.

[7] Elod P. Csirmaz and Laszlo Csirmaz. Data synchronization: A complete theoretical solution for filesystems. Future Internet, 14(11), 2022.

[8] Elod Pal Csirmaz. Algebraic file synchronization: Adequacy and completeness. CoRR, abs/1601.01736, 2016.

[9] John Day-Richter. What's different about the new Google Docs: Making collaboration fast, 2010.

[10] https://drive.googleblog.com/2010/09/whats-different-about-new-google-docs.html, Last accessed on 12 Jan, 2023.

[11] Shimon Even. Graph Algorithms. Cambridge University Press, USA, 2nd edition, 2011.

[12] JiuLing Feng, XiuQuan Qiao, and Yong Li. The research of synchronization and consistency of data in mobile environment. In 2012 IEEE 2nd ICCCIS, volume 02, pages 869–874, 2012.

[13] Rusty Klophaus. Riak core: Building distributed applications without shared state. In ACM SIGPLAN Commercial Users of Functional Programming, CUFP '10, New York, NY, USA, 2010. Association for Computing Machinery.

[14] Zhenhua Li, Christo Wilson, Zhefu Jiang, Yao Liu, Ben Y. Zhao, Cheng Jin, Zhi-Li Zhang, and Yafei Dai. Efficient batched synchronization in dropbox-like cloud storage services. In David Eyers and Karsten Schwan, editors, Middleware 2013, pages 307–327, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[15] Erik Liu. A CRDT-based file synchronization system. Master's thesis, Department of Computer Science, 2021.

[16] Jakub T. Mo´scicki and Luca Mascetti. Cloud storage services for file synchronization and sharing in science, education and research. Future Generation Computer Systems, 78:1052–1054, 2018.

[17] Agustina Ng and Chengzheng Sun. Operational transformation for real-time synchronization of shared workspace in cloud storage. In Proceedings of the 2016 ACM International Conference on Supporting Group Work, GROUP '16, page 61–70, New York, NY, USA, 2016.

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***

ISSN (Online): 2583-648X

IJIRID

International Journal of Ingenious Research, Invention and Development

An International, High Impact Factor, Double-Blind Peer-Reviewed, Open-Access, Multidisciplinary Online Journal

Volume 5 | Issue 5 | October 2026

www.ijirid.in

[18] Andrea Petroni, Francesca Cuomo, Leonisio Schepis, Mauro Biagi, Marco Listanti, and Gaetano Scarano. Adaptive data synchronization algorithm for iot-oriented low-power wide-area networks. Sensors, 18(11):4053, Nov 2018.

[19] Nuno Preguic¸a, Joan Manuel Marques, Marc Shapiro, and Mihai Letia. A commutative replicated data type for cooperative editing. In Proceedings of the 2009 29th IEEE International Conference on Distributed Computing Systems, ICDCS '09, page 395–403, USA, 2009. IEEE Computer Society.

[20] Nuno M. Preguic¸a. Conflict-free replicated data types: An overview. CoRR, abs/1806.10254, 2018.

[21] Yuechen Qian. Data synchronization and browsing for home environments. PhD thesis, Mathematics and Computer Science, 2004.

[22] Bin Shao, Du Li, Tun Lu, and Ning Gu. An operational transformation based synchronization protocol for web 2.0 applications. In Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, CSCW '11, page 563–572, New York, NY, USA, 2011. Association for Computing Machinery.

[23] Marc Shapiro, Nuno Preguic¸a, Carlos Baquero, and Marek Zawirski. A comprehensive study of Convergent and Commutative Replicated Data Types. Technical Report 7506, INRIA, Inria-Centre Paris-Rocquencourt, jan 2011.

[24] Marc Shapiro, Nuno M. Preguic¸a, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In Xavier D´efago, Franck Petit, and Vincent Villain, editors, Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings, volume 6976 of Lecture Notes in Computer Science, pages 386–400. Springer, 2011.

[25] Jian Shen, Member, IEEE, Tianqi Zhou, Debiao He, Yuexin Zhang, Xingming Sun, Senior Member, IEEE, and Yang Xiang, Senior Member, IEEE, "Block Design-based Key Agreement for Group Data Sharing in Cloud Computing", 1545-5971 (c) 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information

[26] Mehdi Bahrami 1 and Mukesh Singhal 2, "A Dynamic Cloud Computing Platform for eHealth Systems", 2015 IEEE 17th International Conference on e-Health Networking, Applications and Services (Healthcom)

[27] Tessema Mengistu∗, Abdulrahman Alahmadi∗, Abdullah Albuali, Yousef Alsenani, and Dunren Che, "A "No Data Center" Solution to Cloud Computing", 2017 IEEE 10th International Conference on Cloud Computing

[28] Zhao Tianhai," The Key of Application Software Service in Science Cloud Computing", 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics

[29] Wang Xiaoyu, Gao Zhengming , "Research and Development of Data Security Multidimensional Protection System in Cloud Computing Environment" , 2020 International Conference on Advance in Ambient Computing and Intelligence (ICAACI)

[30] Yanhong Shang1 and Jing Zhang," Computer Multimedia Security Protection System Based on the Network Security Active Defense Model", Hindawi, Advances in Multimedia, Volume 2021, Article ID 8792105, 9 pages, https://doi.org/10.1155/2021/8792105

[31] L. Zhou, V. Varadharajan, and M. Hitchens, "Cryptographic rolebased access control for secure cloud data storage systems," Information Forensics and Security IEEE Transactions on, vol. 10, no. 11, pp. 2381–2395, 2015.

[32] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, "Secure cloud storage meets with secure network coding," in IEEE INFOCOM, 2014, pp. 673–681.

[33] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," IEEE Systems Journal, pp. 1–10, 2015.

[34] J. Shen, H. Tan, S. Moh, I. Chung, and J. Wang, "An efficient rfid authentication protocol providing strong privacy and security," Journal of Internet Technology, vol. 17, no. 3, p. 2, 2016.

[35] Michał Antkiewicz and Krzysztof Czarnecki. Design Space of Heterogeneous Synchronization, pages 3–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

Content from this work may be used under the term of the Creative Commons Attribution-Non-commercial (CC BY-NC) 4.0 licence. This license allows refusers to distribute, remix, adapt, and build upon the material in any medium or format for non-commercial purposes only, and only so long as attribution is given to the creator. Any further distribution of this work must maintain attribution to the creators. © copyright at IJIRID. ***DOI: 10.5281/zenodo.24124338***