iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.52482

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Secure Decentralised Data Storage and Hiding

Soham Rakhunde[1], Aniket Nikam[2], Shubham More[3], Dhananjay Sangle[4], Prof. Shital Girme[5]
[1, 2, 3, 4, 5]*Computer Engineering Department Pune Institute of Computer Technology, Pune*

Abstract: ***Data is the most valuable thing anybody may have in this technological age, yet encryption alone may not be enough to protect it. Numerous times, sensitive information has been leaked, which has been fatal for numerous organisations. The system works by dividing data into several pieces and dispersing them among peer devices in order to safeguard the data as the complete data will not be present at any single node. The main idea is to leverage a distributed P2P on a local network to store parts of encrypted data across the network and hiding these data chunks by using steganography. A working prototype is constructed to demonstrate the feasibility of the proposed approach.***

Keywords: Information security, distributed systems, steganography, cryptography, hybrid P2P network, AES, Transport Layer Security(TLS)

I. INTRODUCTION

In recent times, there has been a swift evolution of the primary computing paradigms. Centralized computing is progressively advancing through the emergence of Cloud Computing and Data Centers that incorporate some degree of distribution and redundancy. On the other hand, Distributed computing is predominantly realized via Edge Computing and Internet of Things (IoT) applications, which involve smart and interconnected devices such as mobile phones, wearable, sensors, and more. As both computing solutions are addressing a wide range of industrial and personal use-cases, there is a pressing need for other domains to evolve particularly in the realm of cybersecurity. Fog computing is one of the emerging ideas in this field which relies on edge devices to share their resources such as processing power, storage, etc as discussed in [1] as a promising area for developing new solutions in network security, data security, and privacy. Thus, we can leverage fog computing, cybersecurity and steganography to build a secure way of storing data to eliminate single point of failures.

Steganography is used to conceal information in other media like images, video, text, audio, etc. Writing secret information via steganography ensures that only the intended recipient is aware of the presence of the secret message [2]. In the case of conventional steganography, attackers can detect communication via steganalysis. With our proposed solution the attacker would have to steganalyse all images at every node which is computationally very expensive and time consuming, thus not feasible.

Our contribution to this paper is the formulation of a new distributed steganography method.

A technique known as Least Significant Bit (LSB) steganography [3] can be used to conceal data within PNG images.

II. MOTIVATION

In this age of technology, data is considered the most valuable asset anyone can have to safeguard the data; mere encryption may not be sufficient. With the correct procedures, businesses can fortify their networks against threats to data security as mentioned in [4]; nevertheless, current methods are not always adequate. It is important that the system manages the vast volume of data generated in the world and even if one or more machines fail, the system should still work. Our system makes an effort to fulfil these requirements.

III. RELATED WORK

IR. Cogranne et al. [2] have used JPEG image-based steganography and it is based upon the MiPOD (Minimizing Performance of Optimal Detector) technique that improves the estimation of pixel variances. The proposed technique improves upon it by minimizing the performance of an optimal detector for JPEG images. And achieves great results against modern statistical detectors. It has just 0.027 detection rate at 75 Quantization factor using the latest stego-analysis techniques. The approach for steganography of JPEG images that is presented in this research is innovative and notably different from past work that attempts to create a cost function more or less haphazardly. Authors suggest use of hypothesis testing theory to evaluate the statistical performance of the best, most powerful test in the worst scenario when the detector has access to all distribution parameters for each and every pixel. They tested this solution's statistical performance in this "worst-case scenario" to create a data hiding technique that particularly attempts to reduce this performance of Steganalysis tools.

C. Biswas et al. [4] have worked on hybrid cryptography where the symmetric key used for message encryption has also been encrypted using the public key of RSA. Digital signature is created by creating a hash value of the message, which is then encrypted once more with the aid of the RSA public key. This digital signature aids in the receiving end's verification of the message's reliability. In order to create a full communication, the encrypted message, encrypted key, and encrypted digest have all been joined. Utilizing the LSB (Least Significant Bit) steganography method, the entire message has been embedded. This solution thus packages authentication, integrity, and confidentiality.

Jedrzej Bieniasz et al. [5] have worked on distributed steganography linked with the previous work cyberfog security. The authors propose a solution in which stego images are sent over existing BitTorrent P2P networks.The proposed work suggests the use of BitTorrent network as a P2P platform that enables for decentralized multimedia sharing. Due to the structure of this network it is susceptible to be intercepted by malicious third parties as it relies on a very open BitTorrent Network. This work proves to be instrumental in finding a better way of handling addresses with an address generator that relies on relative entropy and n-gram model.

K. S. Puthusseri et al. [6] have worked on fragment replacement strategy.It optimizes the usage of peer resources and ensures symmetry in the system. When compared to a Kademlia-based P2P system, the suggested approach improves symmetry for factors like storage and distance while also dramatically improving file retrievability. However, by utilizing cutting-edge heuristic alternatives, the underlying Pareto Frontier problem of an optimal group selection may be resolved substantially more effectively. In the solution, file retrievability for each additional file is around 30% higher than in Kademlia, demonstrating that availability is not completely sacrificed in the pursuit of system-wide symmetry

IV. REFERENCE ARCHITECTURE

A. Architecture

As mentioned in [5], For communication the system typically has to offer two network services:

1) Sending data between network nodes end to end. There are two main transport layer protocols.

2) Reachability between network nodes. Routing is required because the system's nodes must create a network. A dynamic/static routing protocol could be used to distribute network addresses.

According to Kerckhoff's principle, "Cryptographic systems should be designed to be secure, even if all its details, except for the key, are publicly known." [7]. We followed this principle to build a system which will provide immunity to attacks. Any device on the proposed system would work as a node to provide storage services. The proposed system could be described using an architecture diagram as presented in Figure 1.

a) Data - Multimedia Information that the user wants to store.

b) Data Segmentation - Dividing the data into multiple chunks to store them on peers.

c) Encryption/Decryption - Encrypting the data chunk using AES encryption algorithm. Decrypting the data chunk in case of retrieval.

d) Tracker - Tracks the active peers at any time on the network. It also generates the tracker file i.e the information of where the data chunks are stored.

e) Steganography - Encoding/Decoding the input data into images on PNG using LSB technique.

f) Image - Input image for steganography.

g) Peer Storage - Storage on end devices i.e peers.

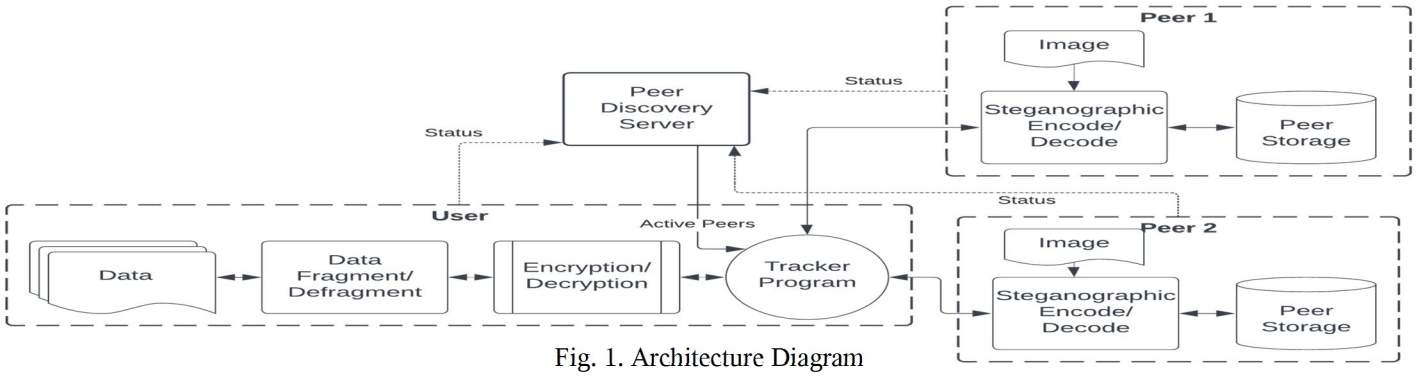


Fig. 1. Architecture Diagram

B. Security Evaluation

In the proposed system, security is provided by data distribution and hiding. The system is reliable as it contains multiple copies of chunks. Apart from these, following are some criteria for security analysis.

1) Availability - information requested should be available.
2) Usability - Information must be accessible in a way that makes it easy to read.
3) Integrity - Integrity is the ability to ensure that a system and its data has not suffered unauthorized modification.
4) Authenticity - the owner of the information should have access to the author of the given information.
5) Confidentiality - keeping in place legal limitations on disclosure and access, as well as measures to safeguard confidential and proprietary information.
6) Ownership - the owner of the information should have access to it.

To summarize:

a) By distributing bits of information amongst end devices, availability is achieved. Independent nodes must be the target of any prospective assault. There is a single operational point of failure (SPOF) that could fail and render the entire system unavailable.
b) In order to guarantee data integrity, the HMAC technique is used on each segment that will be sent [8].
c) An attacker would need to seize control of or make changes to each piece of data transmitted across the end devices in order to imitate the node. A situation like that is extremely unlikely to occur.
d) In the system, confidentiality is realised by default and could be strengthened through implementation. Not all data is accessible to nodes that store data fragments. Before the distribution process, fragments can be encrypted, rendering them unreadable by a possibly hostile node. By default, the system thinks that nodes in the network that aren't told to keep data won't even be aware of it among other objects.
e) Ownership is predicated on the idea that the information owner can ask for access to a particular fragment at any moment.

V. IMPLEMENTATION

The implementation consists of various modules which work in conjunction with each other as a complete system and services that perform operations like store data, and retrieve data. Below each module and services are explained in detail:

A. Key Handler Module

As storing secrets on any node can prove to be catastrophic as the key can be compromised. Thus, we use an open source implementation of a password based key derivation algorithm - Argon2 [9] that uses a Password and a salt (Salt is used to make the key resilient against password dictionary attacks) to generate a 32 byte private key. Due to this the key can be generated every time without the need to store secrets on permanent storage.

B. Cipher Module

Data to be stored is encrypted first using the generated 32 Byte key. We have used the Advanced Encryption Standard (AES) algorithm [10]. AES-256 is used in EAX mode (encrypt- then-authenticate-then-translate) [11] which allows us to prove Confidentiality and Authentication. This is achieved by storing the Cipher, Authenticated Nonce, and the truncated MAC Tag during decryption. The benefit of using AES is that it is hardware accelerated on both Intel & AMD processors.

1) The data is encrypted using AES-256 which takes a 32 byte key and a 16 byte nonce
2) It returns the Cipher and a 16 byte truncated Message Authentication Code (MAC Tag)

C. Data Encoding & Partitioning

The cipher, nonce and MAC needs to be encoded and padded to store them on peers. They can be divided into equal chunks of 32 KB size.

1) Encoding Format:

a) 3 bytes for storing the padding length
b) Padding (if any
c) Ciphe
d) 16 bytes of Nonce

e) 16 bytes of MAC Tag

We divide the encoded data into chunks of size 32 KB. We have decided to store 32 KB to make Stego encode and decode faster and more feasible. The padding is calculated to make the encoded data a multiple of 32 KB.

2) Additional storage used (in Bytes) by encoding:

$$ExtraStorageUsed = 35+(DataSize+35) \bmod (32*1024) \quad (1)$$

D. HMAC Module

Hashed Message Authentication Code (HMAC) [8] is used to ensure Data integrity and authenticity of each chunk. HMAC codes are based upon hashing the data and provide authenticity by using a private key. We are using HMAC with SHA256 as a hash function. Due to this a malicious tracker file cannot route the user to incorrect or malicious data chunks

E. Steganography Module:

After the chunk is received by the peer the data is hidden in multiple images. We have used LSB technique for steganography [3]. Each PNG image can store up to (3*height of image* width of image) bits. This computation is done at peer node and thus has no performance hit on the sender and it adds a few milliseconds of delay to send back the location acknowledgements on the network.

The program encodes the last bit of each color (RGB) value of each pixel of the PNG image to store the data. Using python programming language we implemented the steganographic encode/decode. The PNG images are searched from the home directory on a separate thread and stored as a python unordered set which enables random access and linear access times. Flow chart for LSB is shown in Fig. 2

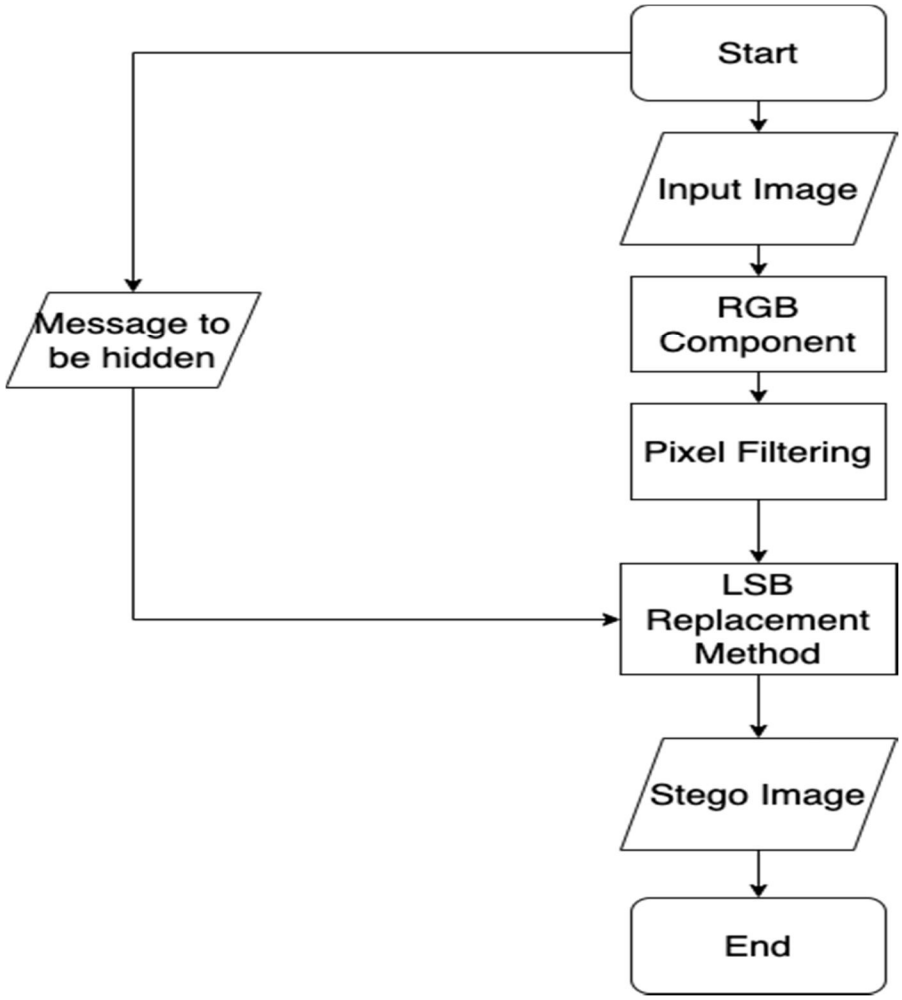


Fig. 2. Least Significant Bit Flowchart

Algorithm for Hiding (Steganography):

1) Choose the cover image and the message to hide.
2) Convert the data that needs to be hidden (H) into binary form.
3) Iterate over each pixel in the cover image and replace the least significant bit of its RGB color value with the corresponding bit from the message until the entire message has been embedded.
4) Save the modified cover image.

Maximum data hiding capacity (in Bits) of an RGB image with a resolution of height*width (pixels):

$$MaxHidingCapacity = 3 * Width * Height \quad (2)$$

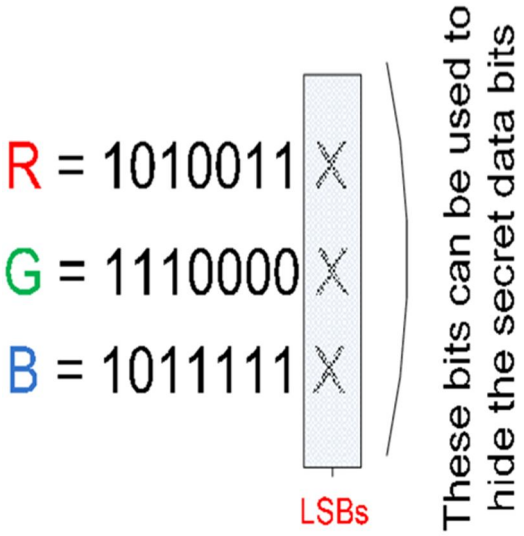


Fig. 3. Least Significant Bit Steganography in one Pixel.

F. Peer Discovery Server:

Peer discovery server enables peers behind NATs (Network Address Translation) to find each other, this is based upon Hybrid P2P paradigm. Every peer connects to the peer discovery server when the program starts. Peer discovery server then can return active peers. In case of data retrieval, among the peers in the tracker file peer discovery server provides the list of active peers.

Peer discovery server can also create isolated networks by making it a compulsion for every peer to enter the Network passphrase to join the network and thus avoid malicious users or potential DDOS attack on the server. Unlike DHT implementations like Kadelima [12], Hybrid P2P allows much faster peer discovery for smaller networks (in this case ideal as organizations can create isolated networks for their use case).

Peer discovery server is implemented in python using the internal libraries like socket,thread,etc. Using socket programming data connection is established. This connection ends when the application gets closed. Each peer connects to this server and can query for active peers. This peer list is stored in memory and used by tracker service or retrieval service.

G. TLS Abstraction Module:

This module is an abstraction layer for sending and receiving the chunks over TLS (Transport Layer Security). It is used by Tracker service and Retrieval service. TLS is very secure as all the data being sent over is ensured encryption, authentication, and data integrity. The sender knows the addresses of its peers, thus it starts a client TLS connection, while the receiver peer does not know the address of sender and thus acts like a server by listening for connection attempts.

1) S (Sender) establishes a connection with a R (receiver peer) using TLS sockets.
2) According to the mode the data is sent or received.
3) In store mode R returns the list of locations of the stego images in which data is stored, while the S waits for this list to be returned and which is later sent back to Tracker service which stores it in tracker file.
4) In retrieval mode the received data is verified with the chunks HMAC (stored in the tracker file), if the received data is corrupted the module repeats requests to retrieve the remaining redundant modules until the correct data is received.

H. Tracker Service (Send Operation):

The tracker service is the main program which handles the communication with peers and handles the chunk queue for send operation. It runs as a separate process with multiple threads to maintain connection with peers parallelly to whom chunks are to be sent and tracks the status of the chunks and its associated peers.

It is also responsible for the generation of a JSON tracker file (similar to torrent file) which stores the HMAC of each chunk and the location of the peer and the stego images(including all redundant copies). This tracker file can be used to retrieve the data from the network.

1) The buffer from data partitioning module & the active peer list from Peer discovery module are taken as input to this program.
2) For each chunk according to its redundancy ratio (number of copies on different peers) a peer (information collected from Peer discovery module's peer list) is scheduled for a connection.
3) This connection is handled by TLS Abstraction Module in new threads spawned for each connection.
4) If the transfer is successful, peer sends the location of the stego image back to the sender.
5) Along with the Chunk number, its HMAC, remote peer location, and stego image location are stored in tracker file
6) If the connection fails or transfer doesn't complete correctly, Step-3 onward are repeated with a different peer.
7) When all chunks are sent successfully the tracker file is saved and can be used to retrieve data using the private key.

I. Retrieval Service (Retrieve Operation)

The retrieval service is the main program which handles the communication with peers to retrieve data. It runs as a separate process with multiple threads to maintain connection with peers parallelly from whom chunks are to be received. It loads the JSON tracker file as input and accordingly connects to the active peers to retrieve data

1) Tracker file is loaded into memory and all the mentioned peers are identified using the MAC address.
2) MAC address list is sent to Peer discovery server and queried for new addresses of the active peers.
3) If all the chunks are retrievable then the program continues else exits with an exception.
4) Threads are spawned to create connections using the TLS abstraction module.
5) After all chunks are received it is re-merged into a buffer and decoded by the Data encoding and partitioning module.
6) Using the derived key the cipher is decrypted and using the MAC tag authenticated. If the decrypted data is authentic it is stored in the home directory.

VI. EXPERIMENTAL RESULTS

A. Experimental Parameters

Parameters	Values
Global Redundancy Ratio	2
Local Redundancy Ratio	2
Nodes in Network 4	4
Data Size	60 KB

B. Parameter Definitions

1) Global Redundancy Ratio : The number of peers on which a copy of a chunk is kept is known as the global redundancy ratio.
2) Local Redundancy Ratio : Local redundancy ratio is the number of copies of chunks stored on a single node.
3) Peers : End user using the application is the peer.

C. Performance

Store Operation: Storing data on a peer device took 661 ms on average for 60 KB file size. Here 'c' represents time required for execution of Encryption, Encoding and setting up Network Connection which can be neglected when compared to Steganography Encoding time.

Store Operation Time = Steg Encode Time + Network Delays + c

Retrieval Operation: Retrieving data without any failures took 844 ms on average for 60 KB file size. Here 'd' represents time required for execution of Decryption, decoding and setting up Network Connection which can be neglected when compared to Steganography decoding time.

Retrieval Operation Time = Steg Decode Time+ Network Delays + d

In case of failure to retrieve, in this case around 700 ms (32 KB steg-decode time) is added for every repeat request of the same chunks as most of the time is used up for steganographic decode.

Failure Case Time(M ax) = Minimum Retrieval Time+ Steg Decode Time ∗ (Local ∗ Global Redundancy Ratios)

Thus we can infer that operation execution time is heavily dependent on the Stenography operations execution time as other operations are relatively much faster and thus doesn't affect the systems execution time by much.

Chunk Size	Encode Time (ms)	Decode Time (ms)
16KB	301	561
32KB	490	684
64KB	1004	2033
128KB	2057	4010
256KB	3993	7960
512KB	7891	15834

CHUNK SIZE VS STEGANOGRAPHY OPERATION TIME

VII. CONCLUSION

Thus we have built an application which improves security by maintaining confidentiality, data integrity, authenticity and data hiding while storing the data on a distributed system. Steganography is used to conceal the fact that data exists on the system which adds an extra layer of security. The storing of data in a distributed manner makes it difficult for attackers to manipulate/access the information. A further stage of this project is to enable sharing of data across the public internet by using a technique called Hole Punching [13].

REFERENCES

[1] S. Sicari, A. Rizzardi. and A, Coen-Porisini. "Insights into security and privacy towards fog computing evolution," Compute Secur., vol. 120. Sep. 2022. An. no. 102822.
[2] R. Cogranne, Q. Giboulot and P. Bas, "Efficient Steganography in JPEG Images by Minimizing Performance of Optimal Detector," in IEEE Transactions on Information Forensics and Security, vol. 17, pp. 1328-1343, 2022, doi: 10.1109/TIFS.2021.3111713.
[3] D. Neeta, K. Snehal and D. Jacobs, "Implementation of LSB Steganography and Its Evaluation for Various Bits," 2006 1st International Conference on Digital Information Management, Bangalore, India, 2007, pp. 173-178, doi: 10.1109/ICDIM.2007.369349.
[4] C. Biswas, U. D. Gupta and M. M. Haque, "An Efficient Algorithm for Confidentiality, Integrity and Authentication Using Hybrid Cryptography and Steganography," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE).
[5] J. Bieniasz, P. Bąk and K. Szczypiorski, "StegFog: Distributed Steganography Applied to Cyber Resiliency in Multi Node Environments," in IEEE Access, vol. 10, pp. 88354-88370, 2022, doi: 10.1109/ACCESS.2022.3199749.Communication Engineering (ECCE), 2019, pp. 1-5, doi: 10.1109/ECACE.2019.8679136.
[6] K. S. Puthusseri, A. Shetty, M. Patil and K. K. Devadkar, "Storage and Availability Aware Fragment Placement for P2P Storage Systems," 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, pp. 0867-0871, doi: 10.1109/ICCSP.2019.8698015.
[7] Petitcolas, Fabien. (2011). Kerckhoffs Principle. 10.1007/978-1-4419-5906-5.
[8] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.
[9] A. Biryukov, D. Dinu and D. Khovratovich, "Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications," 2016 IEEE European Symposium on Security and Privacy (EuroS&P), Saarbruecken, Germany, 2016, pp. 292-302, doi: 10.1109/EuroSP.2016.31.
[10] Daemen, Joan & Rijmen, Vincent. (1998). The Block Cipher Rijndael. Lecture Notes in Computer Science - LNCS. 1820. 277-284. 10.1007/10721064_26.

[11] Bellare, M., Rogaway, P., Wagner, D. (2004). The EAX Mode of Operation. In: Roy, B., Meier, W. (eds) Fast Software Encryption. FSE 2004. Lecture Notes in Computer Science, vol 3017. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-25937-4_25

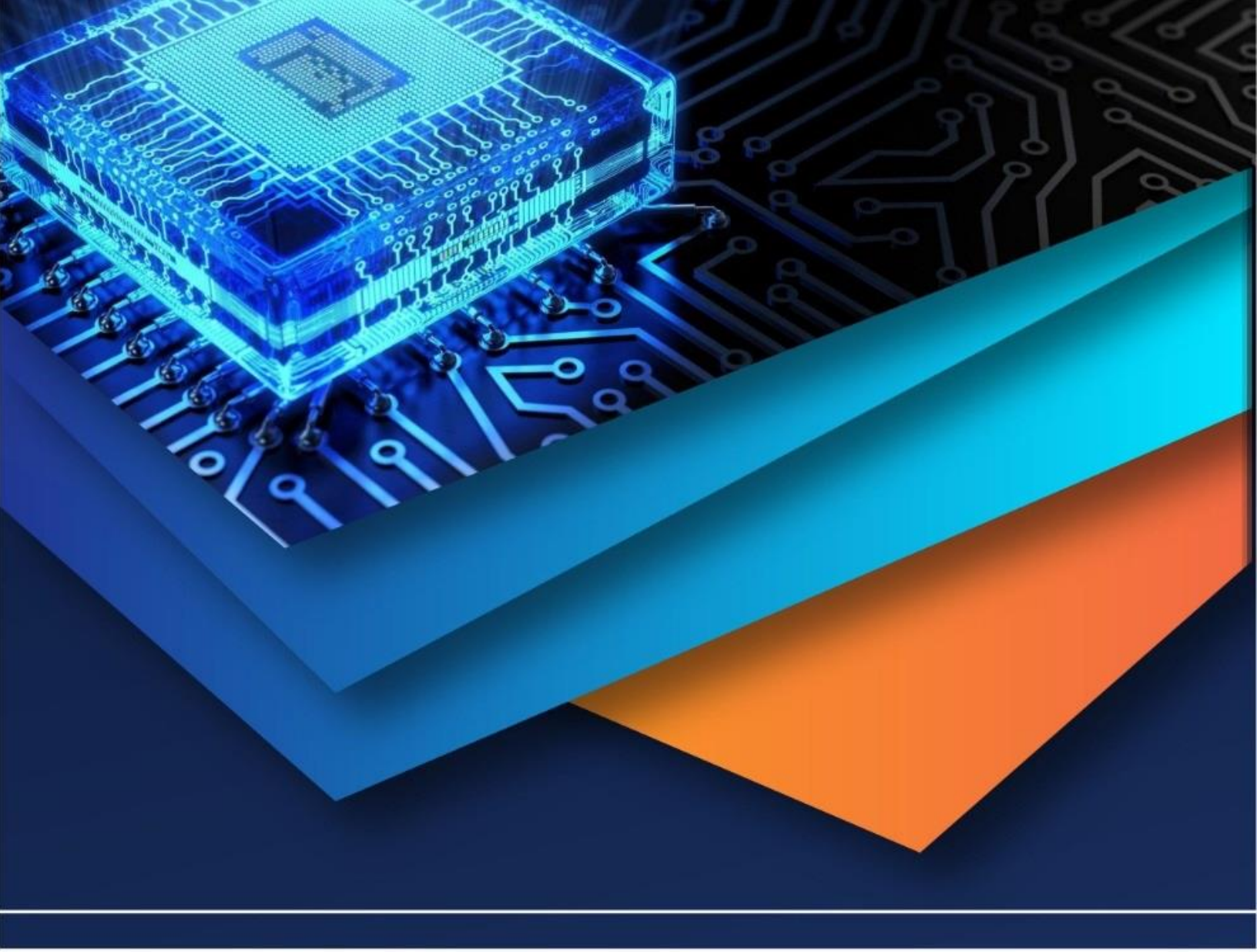
[12] Maymounkov, Petar & Eres, David. (2002). Kademlia: A Peer-to-peer Information System Based on the XOR Metric. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. 2429. 10.1007/3-540-45748-8_5.

[13] Ford, Bryan & Srisuresh, Pyda & Kegel, Dan. (2006). Peer-to-Peer Communication Across Network Address Translators.

[14] X. Liao, Q. -y. Wen and S. Shi, "Distributed Steganography," 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2011, pp. 153-156, doi: 10.1109/IIHMSP.2011.20.

[15] K. Szczypiorski, "StegHash: New method for information hiding in open social networks", Int. J. Electron. Telecommun., vol. 62, no. 4, pp. 347-352, Dec. 2016.

[16] L. Moyou Metcheka and R. Ndoundam, "Distributed data hiding in multi-cloud storage environment", J. Cloud Comput., vol. 9, no. 1, pp. 1-15, Dec. 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY