Multi-tiered Decentralized Logical Network for Secure File Sharing

Akashdeep Dhar[1], Mahij Momin[2], Ankit Sinha[3], Mohit Gurav4, Sheetal Kumar Jain[5]

Department of Computer Engineering and Technology, MIT Academy of Engineering

Abstract

The system aims to establish a decentralized network for protected file sharing. The dynamic network arrangement allows the peer with high data availability and raw performance to preside over and sustain all transmissions. This is done by organizing other fellow peers in a simple tree-like logical arrangement across multiple levels keeping stronger performing devices near root to serve data portions better. Reducing the peer dependency on a single server marks a departure from the archaic client-server model. The parts of the sought data can be acquired from other peers even after disconnection of the actual source. Circulated parts of data are signed specifically so authenticity and integrity are maintained by-design. Instead of encrypting all the portions, the system reduces network traffic and processing load by encrypting only the transfer ledger consisting of encryption keys and portion hashes. This minimizes the chance of eavesdropping and man-in-the-middle attacks by disposing off portions with signature mismatch.

Keywords: *Decentralized network, Multi-tier network arrangement, Self-sustaining connectivity, Synchronous file transfer, Asymmetric key cryptography, File integrity verification, One-time key generation, Ledger-based file exchange*

1. Introduction

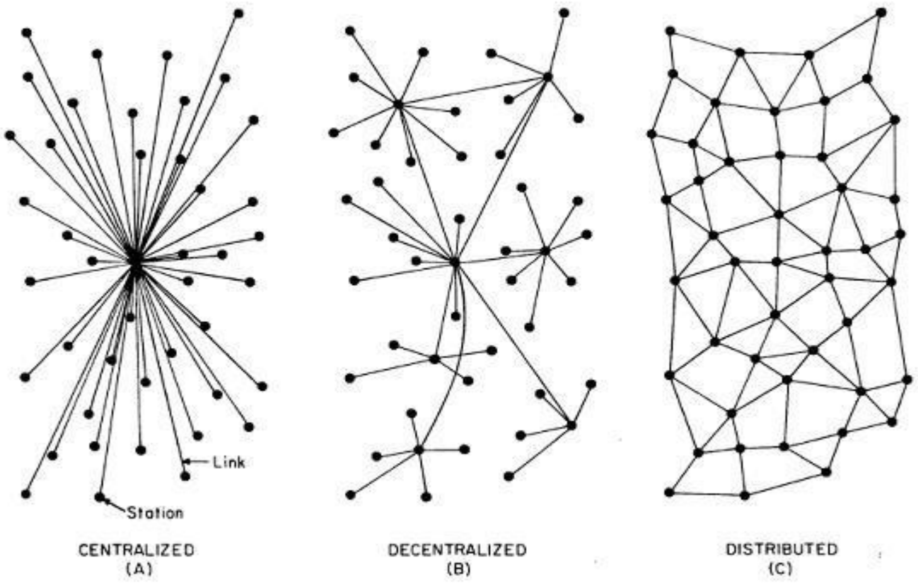

Figure 1 Different Types of Network Models

Information systems and networking are integral parts of today's technology. Connecting multiple information systems together allows for sending available information to others while acquiring those from others. However, depending upon the ownership of the network – the provided data can be challenged for its authenticity, integrity and confidentiality. [1, 2]

1.1. Centralized network

All clients connected to the network are bound to share their data to a central server even in their unwillingness. Fault tolerance in on-point and disconnecting one client from the server would not affect others but a compromised central server will put the entire network at potential privacy risk.[1]

1.2. Decentralized network

All clients connected to the network enjoy privacy as there is no central server to log and eavesdrop the transmissions. [2, 3] Although, failure events might lead to a portion of the network going down for a considerable amount of time as fault detection is difficult. Disconnecting from the network is inconsequential for other devices. [1, 2, 3]

1.3. Distributed network

All clients connected to the network enjoy unparalleled security but at a cost of speed and response times. This network is difficult to manage as every client is a potential server for devices adjacent to it. [1, 3] Disconnecting a client from the network would affect devices in its immediate vicinity causing untraceable fault. [1, 2, 3]

2. Expected improvements

Our research in this field focuses on addressing the following shortcomings of a decentralized network by-

2.1. Improving scalability of the network so that devices can connect and disconnect to the network at will and flexibility on how the logical arrangement of the network is managed. [1, 2, 3]
2.2. Decreasing latency and packet drops during data transmission among two devices situated at dissimilar portions of a decentralized network. [4]
2.3. Removing absolute dependency on a single server by dissolving the serving access to other fellow connected devices so that portions of data can be fetched from anywhere and then integrated.[5]
2.4. Improving security and privacy protection for every device connected so that device or servers lying in path would not be able to listen in and make changes to transmitted information. [1, 2, 3]
2.5. Increasing transferred information quality by maintaining its integrity (using transfer record), authenticity (using digital certificates) and confidentiality (using end-to-end encryption). [6, 7]
2.6. Creating a sustainable decentralized network which will continue on by deputing a peer with highest performance rating and data availability even after the source gets disconnected. [4, 5]
2.7. Diversifying the network by allowing devices situated at the same or distinct tiers of the logical arrangement to share data by one-to-one communication or by broadcasting. [4, 5]

3. Existing systems

3.1. Invisible Internet Protocol

Concept This is an anonymous network layer which bypasses censorship and is supported by volunteer PCs worldwide

Methodology Uses peer-to-peer connection and encrypts user's data with end-to-end encryption. Data is routed through a network of 55,000 computers so tracing is unlikely. Uses ECDSA signatures.

Advantages Perfect privacy is maintained, difficult to trace traffic, bypasses censorship restrictions, re-establishes a new tunnel frequently.

Disadvantages High latency and probability of packet losses, connection is unreliable and droppages are often due to frequent switching.

3.2. Infinit

Concept This is an unlimited file sharing service based on peer-to-peer sharing protocols focused on digital artists.

Methodology Uses peer-to-peer connection with legacy cryptographic algorithms in C++. It has a client application installable supported on multiple OSes for effortless file transfer.

Advantages Moderate privacy protection as all peers are mediated by Infinit, speedy and convenient for one-to-one file sharing.

Disadvantages Network arrangement is static so failures are detectable but frequent, system is obsolete and replaced by a centralized storage platform.

3.3. BitTorrent

Concept This is a peer-to-peer file sharing and communication protocol used to share files over the internet by implementing a 'swarm' of connected peers. [1, 3]

Methodology The file is split into blocks by the seeder and one block per peer is transferred with a peer-to-peer block transfer process. Peer connects to the seeder via information from trackers and continues to leech the seeder for necessary blocks.

Advantages Torrents help reducing stress and dependence on a single server and resourcing blocks from adjacent peers is always faster

Disadvantages Poor seeding increases network bottleneck and vulnerable to BitErrant attacks which alters execution paths

3.4. The Tor Network

Concept This is a free and open-source software which enables anonymous communication by directing the traffic through an overlay network. This network encrypts the data by adding layers to it.

Methodology Uses Onion routing technique which encapsulates the data with encrypted layers so that the decryption at network nodes is just enough to tell about destination and not its contents. An 'exit' allows the data packet to leave the network, decrypted.

Advantages The identity of the user is masked to prevent identification and detection by any third-party and hosting content is possible by bypassing all censoring restrictions.

Disadvantages The exit node can read the data which leaves the network undecrypted which is vulnerable and as TOR does not hide the fact, the users are using it, some sites can detect its use and block the users.

4. Current scope

In the existing systems, the following are the areas that we intend to address with our end of the research by: -

- Reducing latency in transmission by optimizing amount and frequency of data exchange among nodes and decluttering the end-to-end encryption.
- Building a network arrangement which dynamically reorganizes itself to balance network load on every occurrence of device addition or removal.
- Giving a way to maintain and verify integrity of all transferred blocks of a file through the network with as minimum delay due to it as possible.
- Minimizing the dependency of per-node storage and providing limited as well as controlled access to node file systems to other fellow node devices.
- Creating a self-sustaining network by shifting the serving control to the next-most powerful PCs with maximum available data when the primary server disconnects.
- Speeding up transfers by byte-wise block division and hashing of individual files before transmission instead of hashing each and every block for integrity.
- Providing resumability and saving data in transmission by allowing nodes to download only the blocks that have not been previously downloaded in their systems.
- Removing redundancy in peer storage by storing only the parts of data that is not available with the other peers - therefore taking up less local storage space.
- Optimizing the network load by losslessly compressing the blocks one-by-one before they are sent over the network and increasing the transfer rate.
- Disallowing the transfer of suspicious files over the network by letting peers flag content - hence, saving other fellow peers from downloading it.

5. Decentralized Union of Connected Environments (D.U.C.E.)

5.1. Multi-tier Logical Network Arrangement

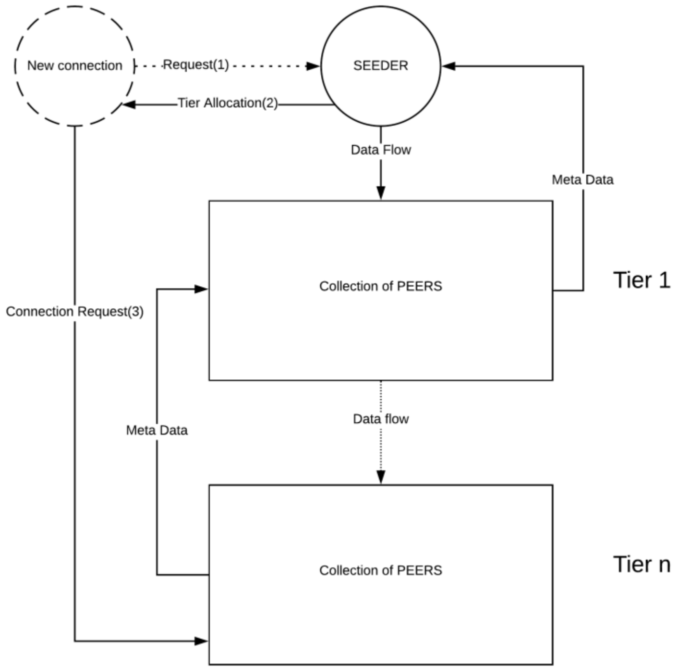

Figure 2 Architectural Diagram

The impact of having cascading meta-data transport is that the effective load on the central seeder to maintain the network structure and to ensure every tier is filled maximum occupancy is reduced exponentially [4, 5]. After the effective file transfer from seeder, the only job remains is to address new connections that want to enter the network. Meta data will decide the effective position of the new connection. Subsequently the meta data will be used to dynamically alter the positions of peers such that peers with more data will be located near to seeder and vice versa.

5.2. Tier architecture

The tier will contain peers connected in mesh form with the constraint of total connections possible within each node. [1, 2, 3] Peers can effectively update the data blocks that are gained and will encapsulate their meta data and cumulatively send it to upper tier. The upper tier will process this data, add their own meta data and forward the meta data. This will continue until the seeder is communicated. Doing so will divide the processing power by number of tiers for maintaining individual meta data for each peer.

5.3. Shadow Clone technique

The main issue will all the peer-to-peer systems is network sustainability [1, 2, 3,5]. So, to address this issue shadow clone technique is introduced. This technique ensures that the network stays alive and can grow even if the main seeder has disconnected but the criteria is that the original seeder must have effectively transferred 100% of its data contents into the network.

6. File Archive Dispersal and Encryption

6.1. One Time Key Service

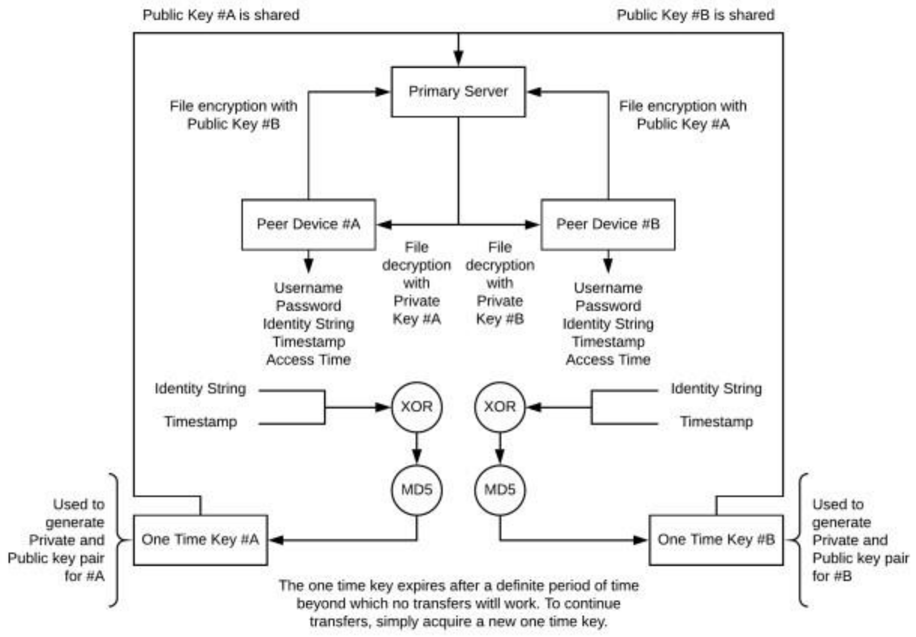
Public Key #A is shared
Public Key #B is shared
Primary Server
File encryption with Public Key #B
File encryption with Public Key #A
Peer Device #A
Peer Device #B
File decryption with Private Key #A
File decryption with Private Key #B
Username Password Identity String Timestamp Access Time
Username Password Identity String Timestamp Access Time
Identity String
Timestamp
XOR
MD5
Identity String
Timestamp
XOR
MD5
Used to generate Private and Public key pair for #A
One Time Key #A
One Time Key #B
Used to generate Private and Public key pair for #B
The one time key expires after a definite period of time beyond which no transfers witll work. To continue transfers, simply acquire a new one time key.

Figure 3 One Time Key Service

Dispersal among multiple peers uses asymmetric key cryptography and a key pair is generated for every user active in the network. Their public key is visible to everyone in the network allowing everyone to send files particularly to them. The private key is stored with the user and cannot be accessed by anyone. Even if the contents shared are trickled down from other peers, they would not be able to view it

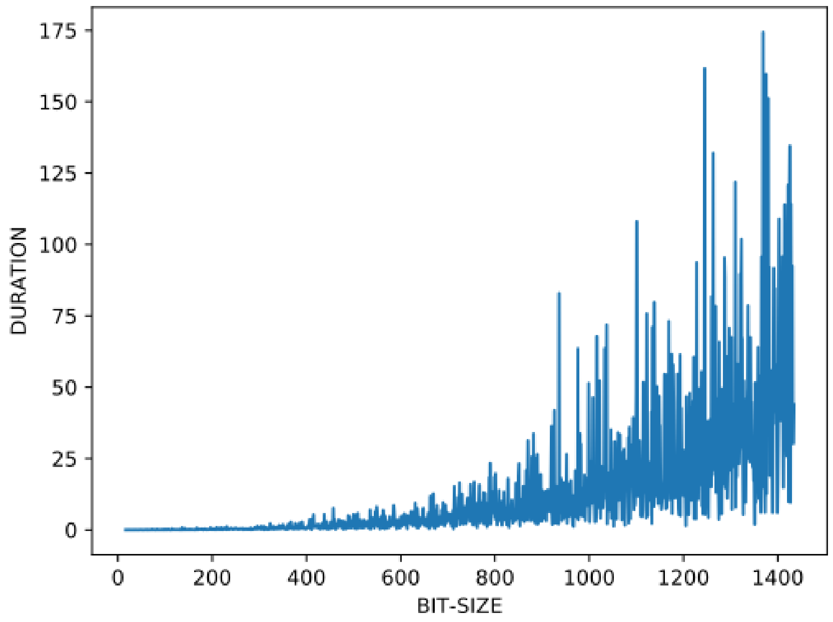

Figure 4 Bit-Size vs Duration plot for key generation

6.2. Encryption of original file using strong symmetric cryptographic algorithm

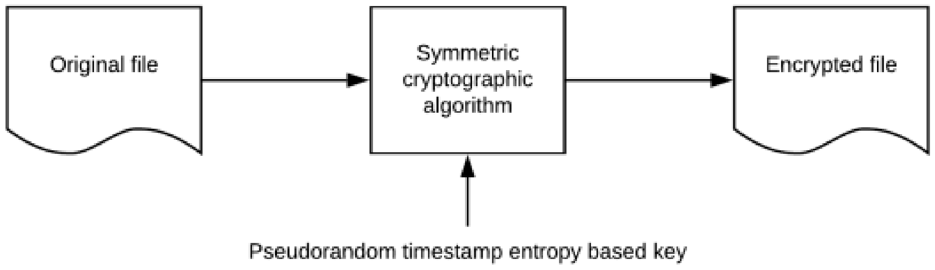
Original file
Symmetric cryptographic algorithm
Encrypted file
Pseudorandom timestamp entropy based key

Figure 5 Encryption of original file using strong symmetric cryptographic algorithm

In order to make the transfers resistant from brute force attacks, the key generation is based on a custom developed pseudorandom timestamp entropy technique.[6] As a time is never repeated, it reduces the possibility of obtaining the same key pair to a probability as low as one is 6 quintillion.[6, 7] Adding that to the time taken to validate attacks and frequent key scrambling, it renders the system uncrackable. [8,9]

6.3. Fragmenting encrypted file with respect to block size or block count

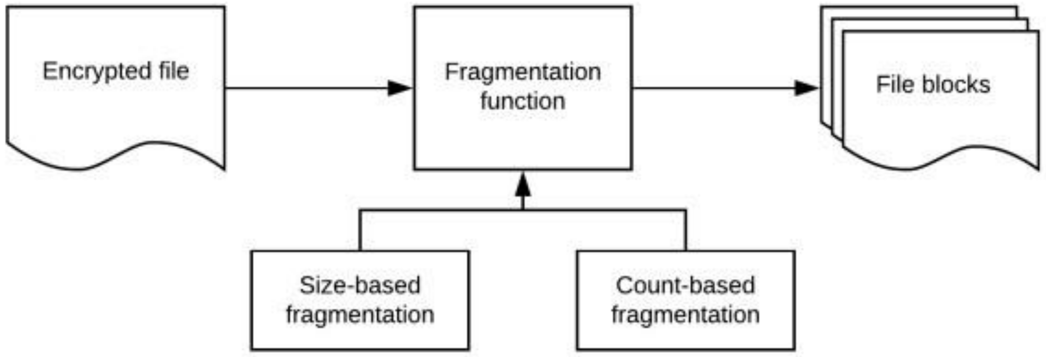

Figure 6 Fragmenting encrypted file with respect to block size or block count

As the sequential reads from the secondary storage devices are considerably slower than random reads, special attention is needed to decide how the file should be fragmented. On faster storages like local SSDs, a size-based fragmentation would be helpful but it might create bigger packets and compromise resumability. Mechanical hard drives can use count-based fragmentation with much smaller packets.

6.4. Hashing every block for integrity verification

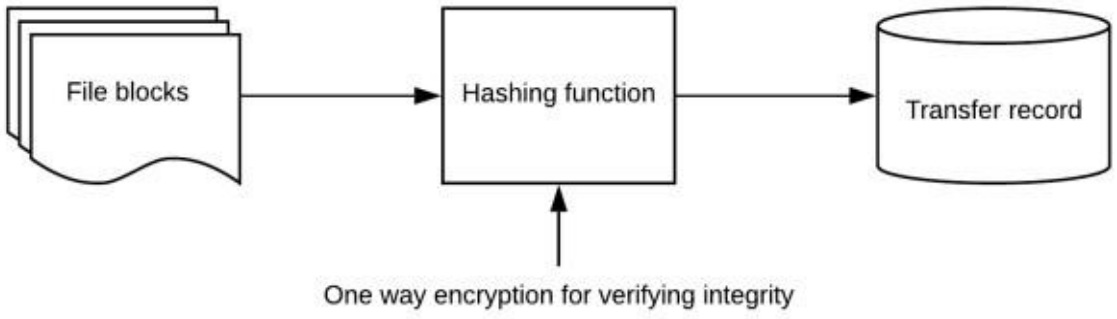

Figure 7 Hashing every block for integrity verification

During fragmentation, every block generated is made to generate a signature with a hashing algorithm - which in turn can be used to verify the integrity of the transferred blocks. As the source can be situated as far as the other end of the network, hashing can help tell if the blocks need to be downloaded again or not. Also, it can tell if the block was manipulated by an unauthorized user during its transit. [6, 7]

6.5. Storing hashes to transfer record and encrypting it with asymmetric cryptography

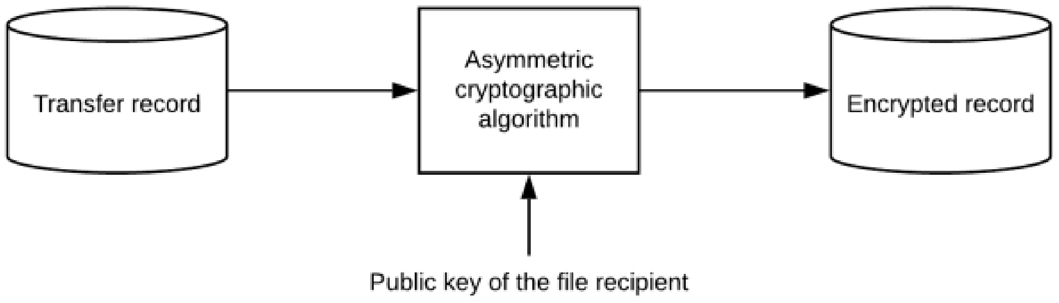
Transfer record
Asymmetric cryptographic algorithm
Encrypted record
Public key of the file recipient

Figure 8 Storing hashes to transfer record and encrypting it with asymmetric cryptography

As much as encryption can be a solution to security issues, doing so on bigger files would lead to considerable read-write latency and slower trickle among peers. [10,11] Instead, only the transfer record is encrypted byte-wise and then transmitted along with the blocks. [7] The transfer record is the only way to reconstruct the file back and failure to decrypt it would infer failure in the entire file decryption - thus keeping the file safe. [6, 7]

6.6. Decrypting the transfer record and retrieving stored hashes from it

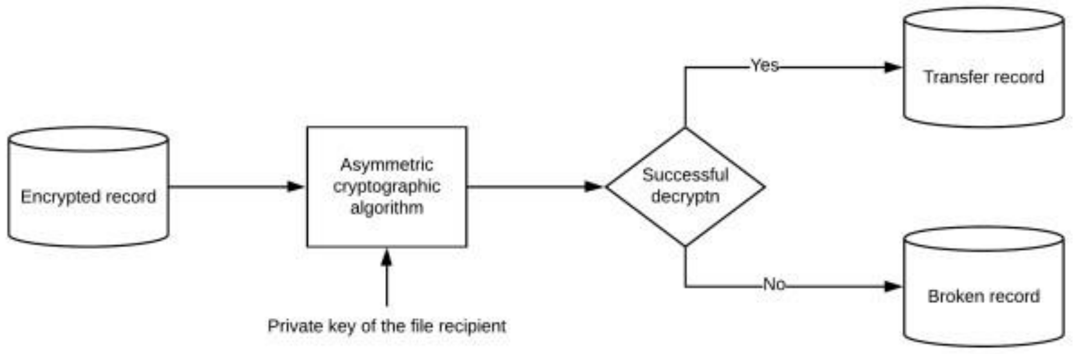

Figure 9 Decrypting the transfer record and retrieving stored hashes from it

On successful decryption of the transfer record, the block signatures can be obtained from it. [6] In cases where decryption fails due to transmission errors or deliberate unauthorized manipulation attempts, the record would be rendered undecryptable and hence, all the file blocks downloaded would be rendered unreadable as there is no way to confirm their integrity and reconstruct to the actual file.

6.7. Verifying all transferred blocks with available hash values

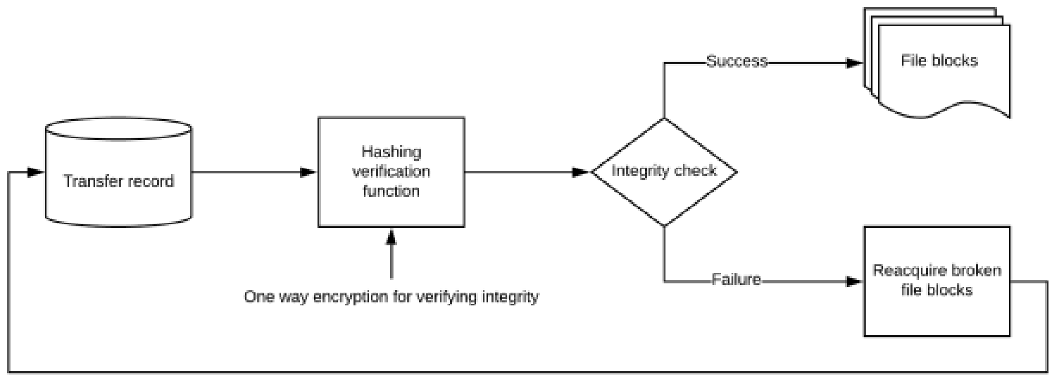
Transfer record
Hashing verification function
One way encryption for verifying integrity
Integrity check
Success
File blocks
Failure
Reacquire broken file blocks

Figure 10 Verifying all transferred blocks with available hash values

A decrypted transfer record would contain distinct signatures of every block. The received blocks would be made to generate their signatures again and both would be compared to check for integrity. [10,11] Any discrepancy in comparison would mean either the block was incompletely received or was modified during transit. These can be downloaded again from the source thereby saving from transferring again.

6.8. Size-wise or count-wise reconstruction of blocks to encrypted file

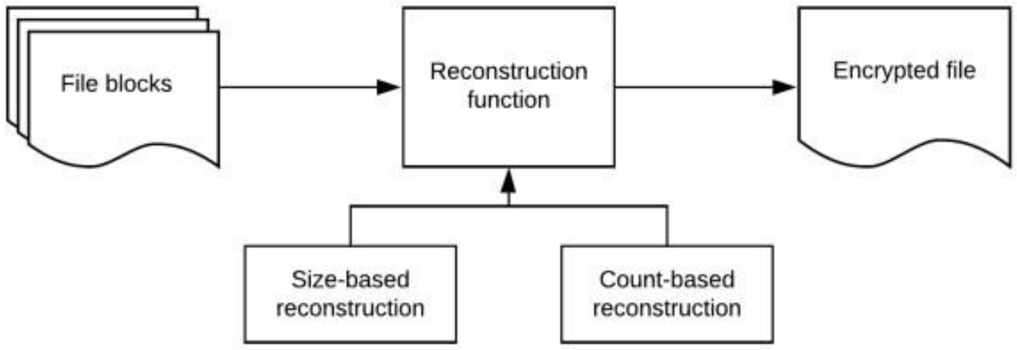

Figure 11 Size-wise or count-wise reconstruction

Based on what kind of deconstruction was performed on the file, the blocks of the file can be reconstructed back into the original file. The two choices are still provided to users for reconstruction as detailed benchmarks on secondary storage specify varied performances across multiple file systems. Testing the system further would cause delay so the choice is ultimately left to the user as to how reconstruction would happen.

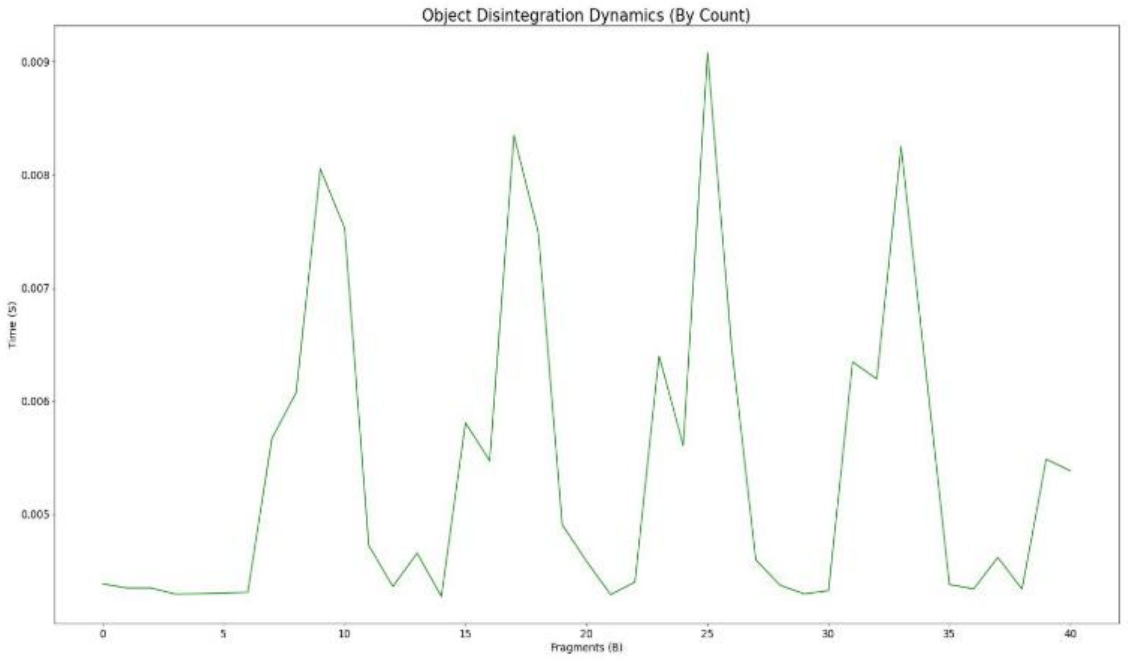

Figure 12 Object Disintegration Dynamics (By Count)

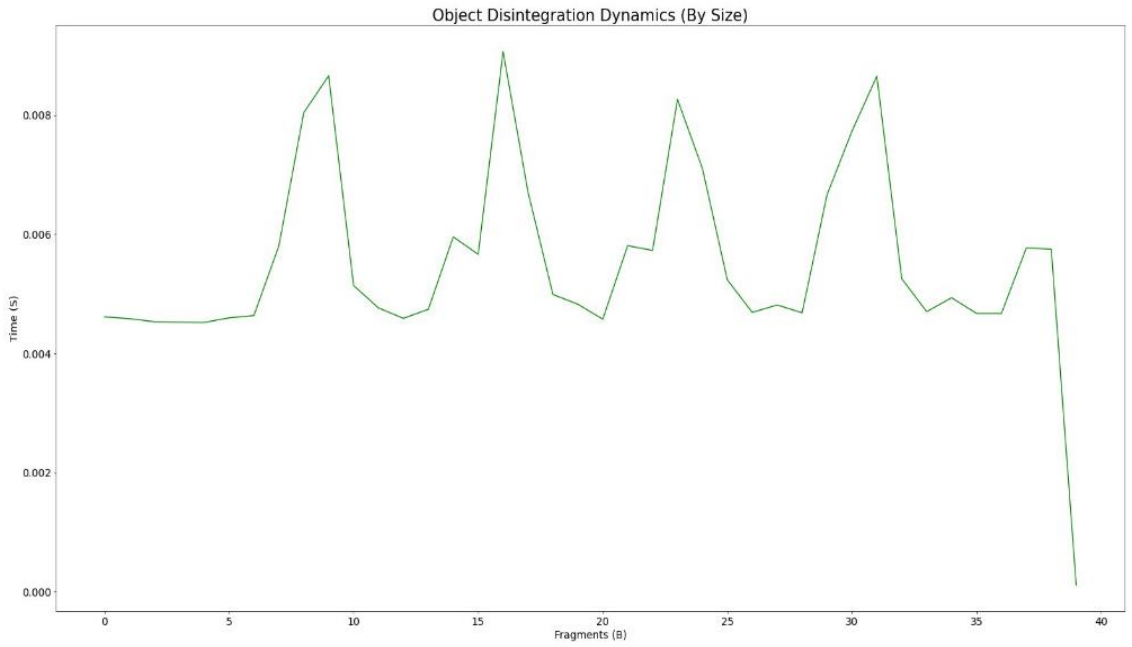

Figure 13 Object Disintegration Dynamics (By Size)

6.9. Decryption from strong symmetric cryptography algorithm to get original file

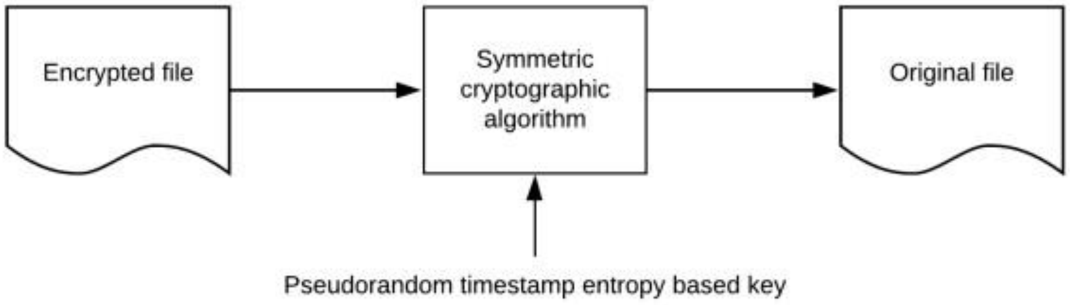
Encrypted file
Symmetric cryptographic algorithm
Original file
Pseudorandom timestamp entropy based key

Figure 14 Symmetric key cryptography decryption

The file encrypted with symmetric cryptography algorithm is decrypted from the key acquired from the transfer record. [8,9] Once the decrypted file is obtained, the split file is retained in order to repurpose it for dispersal among lower tiers. Keeping the file separately would lead to a slight increase in redundancy but would go on to reduce transfer delays, speed up key exchange and circumvent file splitting stages.

7. Discussions

With the impending implementation of the above network architecture, estimates were drawn via simulations as to how well it fares with the existing systems. Following results were drawn.

7.1. Expected speed increase - The use of a custom bucket logic is estimated to provide up to 200% greater transmission speeds. UDP ports are to be repurposed into sending packets of variable sizes over a decentralized network. [1, 2, 3, 10]

7.2. Expected latency reduction - Due to reduced dependence on a single primary server and the distributed nature of all the files shared, there is an estimated rise of up to 400% decline in latency and peer wait times for required file blocks. [1, 4, 5]

7.3. Expected consistency rise - Hashing techniques and maintaining ledger can lead the transmission consistency to an estimated rise of up to 300% - there are lesser chances of failed transfers with segmentation and resuming. [4, 5, 13]

7.4. Expected adaptability rise - The system can be easily implemented up on a bare bones local area network with an estimated rise of up to 100% in scalability - no limits on how many devices can connect and transfer concurrently. [5, 12]

8. Conclusion

The network not only exhibits self-sustainability but is also able to withstand multiple security attacks effectively with the addition of multi-tiering and encryption strategies discussed above. As much as the decentralized network is criticized for weaker security and lesser control, the fact that these systems can end up much faster, fault-tolerant and consistent than the existing ones cannot be denied.

9. Acknowledgements

We express our gratitude to MIT Academy of Engineering for providing various guidelines in writing this paper. We also thank Dr. Shitalkumar Jain (MITAOE) for helping us sort out difficulties and providing insights to our ideas.

10. References

[1] Y. Yang, A. L. H. Chow and L. Golubchik, "Multi-Torrent: a Performance Study," 2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, Baltimore, MD, 2008, pp. 1-8. doi: 10.1109/MASCOT.2008.4770585

[2] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," in IEEE Journal on Selected Areas in Communications, vol. 25, no. 1, pp. 155-169, Jan. 2007. doi: 10.1109/JSAC.2007.070116

[3] Y. Tian, D. Wu and K. Ng, "Analyzing Multiple File Downloading in BitTorrent," 2006 International Conference on Parallel Processing (ICPP'06), Columbus, OH, 2006, pp. 297-306. doi: 10.1109/ICPP.2006.23

[4] H. Harutyunyan and J. He, "A New Peerto-Peer Network," Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), White Plains, NY, 2007, pp. 120125, doi: 10.1109/PERCOMW.2007.9.

[5] J. Sen, "Peer-to-peer networks," 2012 3rd National Conference on Emerging Trends and Applications in Computer Science, Shillong, 2012, pp. iv-v, doi: 10.1109/NCETACS.2012.6203284.

[6] C. Lin, Y. Yeh, S. Chien, C. Lee and H. Chien, "Generalized secure hash algorithm: SHA-X," 2011 IEEE EUROCON - International Conference on Computer as a Tool, Lisbon, 2011, pp. 1-4, doi: 10.1109/EUROCON.2011.5929187.

[7] Xin Zhou and Xiaofei Tang, "Research and implementation of RSA algorithm for encryption and decryption," Proceedings of 2011 6th International Forum on Strategic Technology, Harbin, Heilongjiang, 2011, pp. 1118-1121, doi: 10.1109/IFOST.2011.6021216.

[8] M. Karpinskyy and Y. Kinakh, "Reliability of RSA algorithm and its computational complexity," Second IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings, Lviv, 2003, pp. 494-496, doi: 10.1109/IDAACS.2003.1249613.

[9] S. Chandra, S. Paira, S. S. Alam and G. Sanyal, "A comparative survey of Symmetric and Asymmetric Key Cryptography," 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, 2014, pp. 83-93, doi: 10.1109/ICECCE.2014.7086640.

[10] M. Sivanesan, A. Chattopadhyay and R. Bajaj, "Accelerating Hash Computations Through Efficient Instruction-Set Customisation," 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, 2018, pp. 362-367, doi: 10.1109/VLSID.2018.91.

[11] D. R. Ignatius Moses Setiadi, A. Faishal Najib, E. H. Rachmawanto, C. Atika Sari, K. Sarker and N. Rijati, "A Comparative Study MD5 and SHA1 Algorithms to Encrypt REST API Authentication on Mobile-based Application," 2019 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2019, pp. 206-211, doi: 10.1109/ICOIACT46704.2019.8938570.

[12] Y. Shi, "Data Security and Privacy Protection in Public Cloud," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 4812-4819, doi: 10.1109/BigData.2018.8622531.

[13] L. Dupré and Y. Demchenko, "Impact of information security measures on the velocity of big data infrastructures," 2016 International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, 2016, pp. 492-500, doi: 10.1109/HPCSim.2016.7568374.